IMPRS
for Genome Science
INTERNATIONAL MAX PLANCK RESEARCH SCHOOL

GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Alignment-free Phylogenetic Placement and its Applications

Dissertation
for the award of the degree
"Doctor rerum naturalium"
of the Georg-August-Universität Göttingen
within the doctoral program
International Max Planck Research School for Genome Science (IMPRS-GS)
of the Georg-August-University School of Science (GAUSS)

submitted by
Matthias Blanke

from
Tuttlingen

Göttingen, 2022

Date of the oral examination: February 17, 2023

Zusammenfassung

Das Studium der evolutionären Beziehungen zwischen Lebewesen ist seit jeher ein grundlegendes Forschungsgebiet der biologischen Wissenschaften. In den letzten Jahrzehnten haben große Fortschritte der Sequenziertechnologien für einen enormen Anstieg der Verfügbarkeit von molekularen Sequenzdaten gesorgt. Zeitgenössische Methoden für die Ermittlung von evolutionären Spezies-Beziehungen basieren von daher auf der Bestimmung von Ähnlichkeiten und Unähnlichkeiten zwischen ihren biologischen Sequenzdaten. Das Ergebnis solcher phylogenetischer Studien wird oftmals als Stammbaum oder Netzwerk dargestellt. Während früher jeder Organismus separat sequenziert wurde, ist es heutzutage möglich alle vorhandenen Sequenzdaten aus einer Umweltprobe gleichzeitig zu extrahieren. Dadurch entstehen große Mengen kurzer Sequenzabschnitte deren Ursprungsorganismus unbekannt ist. Ein wesentlicher Schritt der bioinformatischen Aufbereitung solcher Daten befasst sich mit der taxonomischen oder phylogenetischen Identifizierung dieser Sequenzen; aufgrund der großen Anzahl der zugrunde liegenden Sequenzen können dafür keine Methoden zur de-novo Rekonstruktion von Phylogenien mehr eingesetzt werden. Stattdessen stellt die Methode der phylogenetischen Platzierung eine zukunftsfähige Alternative dar: Die phylogenetische Verwandtschaft einer Eingabe-Sequenz wird bestimmt, indem sie direkt in einen bestehenden phylogenetischen Baum aus Referenz-Sequenzen eingeordnet wird. In dieser Arbeit stellen wir eine neue, vielseitige Methode zur phylogenetischen Platzierung vor. Im Gegensatz zu bisherigen Ansätzen zur phylogenetischen Platzierung ist die vorgestellte Methode nicht abhängig von der Verfügbarkeit von alignierten oder assemblierten Referenzen. Stattdessen verwendet unser Algorithmus App-SpaM eine Vielzahl kurzer, nicht-konsekutiver Sequenz-Worte, um eine geeignete Platzierung abzuleiten. Als Grundlage für die Schätzung von Platzierungen dient sowohl die Anzahl der gefundenen Worte als auch eine daraus berechnete phylogenetische Distanz zu allen Referenz-Sequenzen. Wir präsentieren eine umfangreiche Evaluation, die App-SpaM mit anderen Programmen zur phylogenetischen Platzierung hinsichtlich seiner Genauigkeit und Effizienz vergleicht. Unsere Analysen zeigen, dass die Präzision von App-SpaM vergleichbar ist mit der Präzision von existierenden maximum-likelihood Methoden, wobei es zwei bis dreimal schneller ist als diese. Im Anschluss stellen wir weitere Versionen des Algorithmus vor, insbesondere erweiterte Platzierungs-Heuristiken, ein Maß für die Unsicherheit der abgeleiteten Platzierungen und die Verwendung von stochastischen Stichproben, um eine Skalierbarkeit auf langen Referenz-Sequenzen zu gewährleisten. Zusätzlich diskutieren wir mehrere Anwendungsfälle der phylogenetischen Platzierung mit App-SpaM und zeigen deren Umsetzbarkeit an exemplarischen Experimenten. Dabei betrachten wir primär die iterative Ergänzung von bestehenden Stammbäumen, sowie die Detektion von Gen- oder Speziesausreißern.

Abstract

The study of the evolutionary interrelations of living organisms has been at the heart of biological sciences all along. A revolution in sequencing techniques in the past decades has caused a massive increase in molecular sequence data. As a result, contemporary methods assess evolutionary relationships between organisms by quantifying the degree of similarity between their biological sequence data. The discovered relationships of phylogenetic studies are commonly represented and visualized by phylogenetic trees or networks. Traditionally, sequences have been extracted from single organisms; however, recent technological progress has enabled the retrieval of sequence data directly from environmental samples. In doing so, large numbers of short sequencing reads arise that may originate from all organisms present in the respective environment. One major subsequent objective is the taxonomic or phylogenetic identification of those sequencing reads. However, longstanding maximum-likelihood-based de-novo phylogeny reconstruction methods are limited in their applicability by their computational demands; typically, they cannot be applied when the available molecular sequences are present in great numbers or are of great length. Fortunately, phylogenetic placement offers a unique approach to identify large sets of query reads within their phylogenetic context by inserting them into an existing phylogenetic tree comprising a set of reference sequences. Here, we present a new alignment- and assembly-free approach to phylogenetic placement, the **A**lignment-free **p**hylogenetic **p**lacement algorithm based on **Spa**ced-word **M**atches (App-SpaM). App-SpaM extracts short, non-contiguous subwords to detect homologies between the query and reference sequences, a method known as the spaced-word matches approach. It counts the number of such words and utilizes them to infer the average number of nucleotide substitutions between each read and each reference sequence. Then, it uses fast heuristics to infer a suitable placement position within the reference tree. We assessed how App-SpaM compares to existing algorithms for phylogenetic placement with respect to accuracy and computation speed in a comprehensive evaluation. We demonstrate that App-SpaM is on par with maximum-likelihood-based algorithms on metataxonomic data sets. In addition, App-SpaM is two to three orders of magnitude faster than the next fastest programs while its memory demands stay low. We extensively discuss App-SpaM's advantages and drawbacks and propose several additional features to improve upon its original version: For this, we evaluate a set of novel placement heuristics, the use of sampling techniques to allow an improved scalability with the length of the reference sequences, and a measure for the uncertainty of proposed placement positions. Subsequently, we present a variety of novel use cases of phylogenetic that are made uniquely possible by App-SpaM's versatility with respect to its potential input data. These applications include, in particular, the iterative augmentation of existing species trees by means of phylogenetic placement and the screening for outlier genes or species prior to phylogeny reconstruction.

Acknowledgements

My thanks go to Burkhard Morgenstern for his supervision, support, and encouragement during the last four years. Burkhard is an excellent scientific advisor and it was my honor to work alongside him and receive his thought-out advice and helpful guidance. Furthermore, he always supported me personally and endorsed a healthy balance between work and private life that enabled the satisfying work environment I experienced. I will remember my time as a PhD student happily and ascribe this largely to Burkhard.

My thanks go to the members of my thesis advisory committee Johannes Söding and Christoph Bleidorn. With his thoughtful and constructive criticism, Johannes elicited my scientific diligence, expanded my knowledge, and improved my work substantially. Christoph incited my interest in the biological realm, especially the field of biodiversity, which also led to my uptake of new personal hobbies. The obtained perspectives will outlast this work.

My thanks go to the members of our working group, specifically Peter Meinicke for prolific discussions, Thomas Dencker for his valued input on being a PhD student and software design, and Rasmus Steinkamp and Britta Leinemann for their swift help regarding all matters. I also thank the reliable and motivated students I had the pleasure working with: Sarah Wendte, Clara Gisela Köhne, Dark Engel, Rebecca Fee Dietlinde Regendantz, and Patricia-Franziska Römer.

My thanks go to my friends. Most of all, I thank Leo for his continued support in all phases of life, for challenging my work and working practices, for proofreading this manuscript, for joint vacations, rare visits, frequent calls, and for being there in difficult situations. I also thank Birte, Marlene, Salomea, Stefan, and Tobias for nurturing our long-standing friendships even though I have been unresponsive at times. Moreover, I thank Immo, Justus, Malte, and all the wonderful people at Bouldern in Göttingen who share the enjoyment of our common sport.

My thanks go to my family: I thank Tilman for discussions that covered every aspect of life there is to cover, for his many invitations I declined, and for his honest disinterest in my work that puts things into perspective—after all, there is more to life, isn't it? I thank Ulrike for her amiable understanding of personal matters, her kindness and uplifting attitude, and her unwavering support in all stages of my studies and life. I thank Eberhard for his empathetic and congenial participation in my work, our soothing hikes outdoors, and our endless debates on scientific and unscientific matters that continue to widen my horizons. I could not be more fortunate than to have this loving family.

Lastly, my thanks go to Adrienne who filled the last years with the life they needed. Thank you for your energy and excitement in mundane routines, for the countless tours in nature, for being there in challenging hours, and for sharing with me this ongoing adventure of life. There is a great comfort in knowing you by my side.

Contents

Chapter 1

Introduction

Life on Earth is complex and diverse. It is estimated that the first forms of life existed in hydrothermal vents on the ocean floor more than 3 billion years ago [1]. Since then, over thousands of millions of years [1, 2], a multitude of different forms of life have emerged and vanished again. It is extremely difficult to estimate the number of unique organisms that live today, and current estimates range from 8.7 million species [3] up to 1 trillion species [4]. This plethora of organisms varies in their size and shape, in their abundance and abode, and in their metabolism and morphology. However, all organisms are also connected to one another: The fundamental information that defines all organisms is encoded in their DNA, the hereditary material. DNA is passed on from parent to offspring or shared via a variety of complex mechanisms between living organisms [5]. From the smallest bacteria to the largest mammal, organisms share a common history through evolution and are related to one another with regard to their genetic content. Observing and measuring such relationships between organisms from all areas of the tree of life has always been an essential part of biological analyses. The study of evolutionary relationships between organisms is called *phylogenetics*. There are a wide variety of inference methods that study phylogenetic relationships, and the results are commonly represented and visualized as *phylogenetic trees*.

In early phylogenetic research, phylogenetic trees were created based on the apparent visual characteristics of a species, its *phenotype*. The phenotypes of organisms were used to group species together and determine their degree of relatedness; by now, however, phylogenetic analyses predominantly study the genetic material of organisms, the *genotype*. Thus, phylogenetic inference today is performed by analyzing the similarity of biological sequences. On the one hand, this change is motivated because the genotype offers much more detailed information about the evolutionary history than the phenotype. On the other hand, this process was also fueled by the drastic decrease in sequencing costs and the associated rise in available sequencing data. The development of sophisticated sequencing techniques also allowed the retrieval of DNA directly from environmental samples instead of only from a single species. This area of research has been termed *metagenomics*. The most common sequencing data in metagenomics are large collections of short DNA fragments, called *reads*. Metagenomics is especially suitable for studying the composition of bacterial communities, but several related fields of application have emerged within metagenomics [6].

Advances in sequencing technologies also entailed larger computational demands: Early algorithms for sequence comparison are no longer suited to appropriately process the amounts and types of sequence data that arise in metagenomics. Reasonable processing time and memory requirements are important characteristics of algorithms to be able to handle metagenomic data sets. One of the most important tasks in many metagenomic studies today is to identify which species are present in an environmental sample [7]. This task is commonly referred to as *read assignment*. A multitude of programs have been developed to perform this task on a taxonomic basis. Short reads are queried against large annotated reference databases, and the taxonomic labels of the most similar reference sequences are transferred to the query reads. Nonetheless, this approach may fail for query reads if no similar references are present in the database or if the taxonomic annotation of database entries is faulty. Another approach to read assignment is via the phylogenetic context of the reads. This procedure is known as *phylogenetic placement*. In phylogenetic placement, all query reads are assigned to a specific position in an existing phylogenetic tree of a set of reference sequences. Thus, the resulting positions in the tree can not only be used to infer taxonomic labels for the queries, but also their phylogenetic relation to all reference sequences. So far, information obtained through the process of phylogenetic placement is predominantly used synonymously with taxonomic read assignment in metagenomic studies.

1.1 Motivation

Phylogenetic placement is a valid alternative to taxonomic read assignment to identify species membership for short reads from metagenomic experiments. However, with todays speed and variety of sequencing data, new use cases for phylogenetic placement emerge. The algorithms initially developed for phylogenetic placement depend on time-intensive multiple sequence alignments and are limited with respect to their applicability to large data sets with many references or many query reads. Additionally, the dependence on multiple sequence alignments restricts the potential input data types and, by this, the applications in which phylogenetic placement can be used in the first place. The reference sequences need to be assembled, which requires high sequencing coverage or long reads, as well as laborious assembly pipelines. Furthermore, the reference sequences need to be sufficiently short to be able to compute a multiple sequence alignment. This is not only due to the computational demands of sequence alignments but also because longer sequences may exhibit rearrangements, duplications, or other evolutionary events that alter the overall sequence order and hinder the creation of meaningful multiple sequence alignments. Both requirements are regularly not met, for example, in low-coverage whole genome shotgun sequencing studies [8, 9]. As a result, phylogenetic placement has only been used in a similar manner to taxonomic read assignment, with the overall goal of classifying reads from metagenomic experiments in their phylogenetic context to derive a taxonomic label.

Although it has been claimed that phylogenetic placement is more accurate than taxonomic read assignment, especially when there are no close reference sequences [10], the use cases of phylogenetic placement are more diverse. An algorithm that does not depend on multiple alignments comes with several benefits: First, it accepts nearly arbitrary reference and query sequences as input. This includes, for example, sets of unassembled reads from single genes or whole genomes that are utilized as reference sequences, or query reads that are arbitrarily long. Furthermore, such an approach would enable a multitude of use cases for phylogenetic placement at the interface of metagenomics and phylogenetics other than read assignment. Potentially, it could perform rapid identification of new metagenomic reads, binning of scaffolds, fast integration of new species into existing phylogenetic trees, or detection of outlier genes or species with respect to their evolutionary history.

For these reasons, we implemented an assembly- and alignment-free algorithm for phylogenetic placement, called the alignment-free phylogenetic placement algorithm based on spaced-word matches (App-SpaM). App-SpaM is based on filtered spaced-word matches [11] to estimate the average number of nucleotide substitutions between every query and reference sequence. Subsequently, different placement heuristics are used to insert a given query read into the phylogenetic reference tree based on its estimated distances to each reference sequence. We studied the accuracy of App-SpaM for the task of taxonomic read assignment across a broad variety of potential scenarios in comparison to a variety of other alignment-based programs [10, 12–15]. Moreover, we examined the benefits and drawbacks of different algorithmic strategies and investigated additional potential use cases for alignment-free phylogenetic placement. These include the iterative augmentation of phylogenetic trees, and species and gene outlier detection in phylogenetic reconstruction studies. We propose techniques to perform these tasks based on the placement information of DNA sequences created with App-SpaM or other alignment-free phylogenetic placement algorithms and assess their potential use in biological studies.

1.2 Structure and Overview

We present the foundations for all of our work in Chpt. 2. This includes relevant basics of genetics, alignment-based and alignment-free sequence comparison methods, foundations of genomics and phylogenomics, an introduction to metagenomics, and lastly the concepts of read assignment and phylogenetic placement. The foundations are followed by a detailed description of App-SpaM in Chpt. 3. App-SpaM performs alignment-free and assembly-free phylogenetic placement with high accuracy that is on par with other alignment-based software tools. Furthermore, it enables the use of phylogenetic placement for a range of data sets with arbitrary sequence lengths and versatile sequence types such as draft genomes as references. The chapter includes a comprehensive evaluation of App-SpaM on simulated and real-world data sets and highlights its shortcomings. Chapter 4 presents a variety of variations of App-SpaM which include, in particular, different placement heuristics, the use of other evolutionary models to estimate sequence similarity, different strategies to sample spaced words to enhance its runtime, and approaches to assess the uncertainty of inferred placements. We evaluate and discuss relevant shortcomings of and potential future work on these ideas in detail. Chapter 5 presents how phylogenetic placement may be used to iteratively update phylogenetic species trees with new species based on individual gene placements. We employ App-SpaM to augment existing trees with multiple additional species and evaluate the topological accuracy of resulting trees. Additionally, we demonstrate how the detection and removal of outlier genes or species can improve the accuracy of resulting tree topologies. Subsequently, two smaller side projects are presented in Chpt. 6: First, we apply alignment-free methods to reconstruct the phylogeny of eight Old World monkeys and second, we assess how the Simon's congruence behaves for simulated DNA sequences. Lastly, Chpt. 7 contains a comprehensive discussion about the advantages and drawbacks of our presented techniques and how our analyses fit into the overall metagenomic and phylogenomic research context of today.

Chapter 2

Foundations

In the field of biology, *taxonomy* refers to the study of arranging organisms into hierarchically organized groups with respect to their similarity. Taxonomy also aims to name the defined groups and to specify their characteristics. The latest taxonomy groups living organisms into three major *domains* of life, namely, Archaea, Bacteria, and Eukarya [16]. Organisms without a cell nucleus are referred to as *prokaryotes* — the two domains of Archaea and Bacteria contain all prokaryotes. On the contrary, the domain Eukarya contains all organisms that do have a cell nucleus, the *eukaryotes*. The cell nucleus encloses the genetic material of the cell and separates it from the surrounding *cytoplasm*. The cytoplasm contains several other subunits of the cell that perform different functions, called *organelles*. *Mitochondria* are organelles that act as energy suppliers of eukaryotic cells; plant cells have further organelles referred to as plastids, such as the chloroplasts. Mitochondria and plastids contain their own genetic material that is independent of the one in the cell nucleus.

The vast majority of all organisms are classified as prokaryotes, and only a small proportion of organisms are classified as eukaryotes [17]. Furthermore, most prokaryotic life is unknown today due to the sheer number of existing organisms and the difficulty in finding and identifying them. Below the three domains of Archaea, Bacteria, and Eukarya, living organisms are grouped into the seven *kingdoms* of Bacteria, Archaea, Protozoa, Chromista, Plantae, Fungi, and Animalia [18]. Subsequently, the organisms are hierarchically grouped into the taxonomic categories *phylum*, *class*, *order*, *family*, *genus*, and *species*. Creating a taxonomy is only meaningful because organisms pass on and exchange their genetic material, which causes the similarities and differences that are used for grouping and delineating species. The field of *genetics* studies how, where, when, and by which processes heredity takes place in organisms.

2.1 Genetics

Deoxyribonucleic acid (DNA) is the genetic material in both prokaryotes and eukaryotes. The DNA contains all genetic information necessary for the complete functioning of an organism, including information on growth and reproduction. DNA was first isolated in 1869 [19] and in 1944 it was demonstrated to be responsible for heredity [20]. Figure 2.1 shows a schematic representation of the DNA structure. The DNA molecule consists of two strands that are intertwined in a double helix structure [21]. Each strand is a chain of consecutive nucleic acids, also called *nucleotides*, each formed from a sugar, a phosphate group, and one of the nucleobases *adenine* (`A`), *cytosine* (`C`), *guanine* (`G`), and *thymine* (`T`). Based on their chemical structure, `A` and `G` are classified as *purines* and `C` and `T` are classified as *pyrimidines*. In each strand, the sugar of one nucleotide is connected to the phosphate group of the next one. The two strands are complementary: the same pair of nucleotides are always opposite to each other

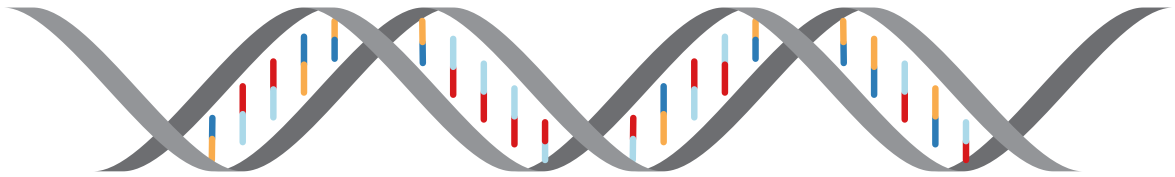


Figure 2.1 – Schematic representation of the double helix structure of DNA. The helix backbone of alternating sugar and phosphate molecules (grey) acts as a scaffold for the nucleobases of each nucleotide (colored bars). The same two nucleotides always form a base pair, and thus, the same colors are always opposite to each other: adenine (orange) pairs with thymine (blue) and cytosine (cyan) pairs with guanine (red). The genetic information of an organism is encoded in the sequence of nucleotides.

on both strands and form a *base pair* (bp). Hydrogen bonds between the complementary nucleotides stabilize the DNA molecule. `A` and `T` form a base pair with two hydrogen bonds, and `C` and `G` form a base pair with three hydrogen bonds. The complementary structure of DNA implies that one strand can be reconstructed by knowing the other and vice versa.

In eukaryotes, DNA inside the cell nucleus is packaged into *chromatin*. Several chromatin complexes form multiple linear *chromosomes*. In most prokaryotes, DNA is packaged into a single circular chromosome instead. The complete genetic material of an organism is called its *genome* and the study of the structure and function of genomic regions is called *genomics*.

The genetic information of the DNA is encoded by the sequence of the nucleotides. DNA regions that code for the synthesis of biological products are called *genes*. Genes are also referred to as *coding* regions, while other parts of the DNA are called *non-coding* regions. During the process of *gene expression*, genes are first *transcribed* to *ribonucleic acid* (RNA). RNA has only a single strand, compared to the double-stranded DNA, and folds onto itself. Furthermore, the nucleotide *uracil* (`U`) is used instead of thymine. There are many different kinds of RNA that serve a variety of biological purposes, for example, in the regulation and expression of genes [22, 23]. However, in most cases, the transcribed RNA serves as *messenger RNA* and is used as a template for the construction of *proteins* during the process of *translation* in *ribosomes*. Here, three consecutive nucleotides, called a *codon*, are translated into one *amino acid*. Amino acids are organic compounds that come in a great variety: The DNA codes for 20 different amino acids, but approximately 500 amino acids are naturally occurring. With 4 different nucleotides there are $4^3 = 64$ different possible codons for only 20 amino acids. As a consequence, multiple codons encode the same amino acid. Thus, any DNA sequence can be unambiguously translated into a sequence of amino acids, while the converse does not apply.

When a sequence of nucleotides is translated into amino acids, the starting position of the translation is essential. Shifting the starting position by a single nucleotide in either direction will result in different amino acids because the shift results in different codons. Six different *reading frames* of a DNA sequence are possible when considering both strands of a DNA. Typically, there is only a single biologically relevant reading frame for each DNA sequence called its *open reading frame* (ORF). The translated amino acids form a chain, a so-called *polypeptide*, that in turn forms a protein. Proteins ensure the proper functioning of organisms and are responsible for a wide variety of vital tasks.

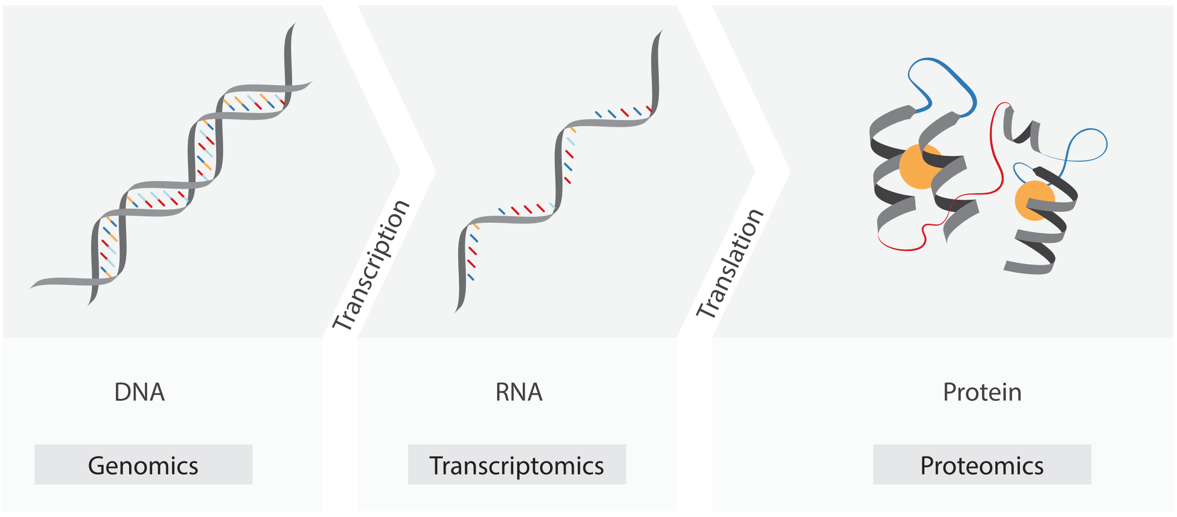


Figure 2.2 – The relationship between different fields of genome science: Genomics studies the DNA of the whole genome, transcriptomics the transcribed RNA of the transcriptome, and proteomics investigates translated proteins. Furthermore, the field of metabolomics (not shown) studies all metabolites within a cell or organism.

The proportion of coding to non-coding regions differs greatly between organisms. The estimated proportion of coding regions in the human genome is estimated to be approximately 1.5 percent of the total genome [24]. The coding regions encode an estimated number of 20 000 genes [25], however, there are different estimates that also depend on the implied definition of a gene [26]. On the contrary, most prokaryotic genomes are densely packed with genes, and the proportion of coding regions is often greater than 85% [27].

Similarly to genomics studying the genome, the complete collection of RNAs of a system is called its *transcriptome* and is studied in the field of *transcriptomics*. The *proteome* of an organism is its collection of proteins and is studied in the field of *proteomics*. Figure 2.2 shows a schematic representation of the relationship between genomics, transcriptomics, and proteomics. Thus, biological sequence data is available as either DNA, RNA, or as a protein sequence. All genetic material within an organism is called the *genotype* of the organism. Through transcription, translation, and further biological processes, the genotype is directly responsible for the visible features that define an organism, the *phenotype*. Organisms are related to another with respect to their DNA content.

The DNA within a cell is subject to change through a variety of biological processes. This results in genetic variability between cells, and accordingly between organisms and species. In sexual organisms, *recombination* is a major source of genetic variability: During reproduction, genetic material is transferred from ancestors to offspring. The DNA of the offspring is a recombination of the corresponding chromosomes of the two parental cells. In organisms that have more than two copies of each chromosome, so-called polyploid organisms, the number of possible genetic recombinations is amplified. DNA from organelles, such as mitochondrial DNA, is transferred directly from the maternal organism and is not subject to recombination. In asexual organisms, the offspring receives an almost exact copy of the parental genome. Here, *horizontal gene transfer* (HGT) is one of the main processes of genetic exchange between cells: During an HGT, DNA or RNA is transferred from a living cell to another by a variety of mechanisms [5, 28]. HGT occurs mostly in prokaryotes; however, it also exists in certain eukaryotes [29, 30], and between pro- and eukaryotes [31]. Another important source of genetic variability is through *mutations*. Mutations are alterations of DNA or RNA within a cell. They usually occur through errors during the replication of a sequence, but many other reasons exist such as error-prone DNA repair processes [32, 33]. The alteration arises as an *insertion*, a *deletion*, or a *mutation*. Insertions occur when one or multiple new nucleotides are incorporated into the sequence. Deletions occur when one or more nucleotides are removed from the sequence. A mutation is the exchange of one or more nucleotides for other nucleotides. *Point mutation* are mutations of single nucleotides in the sequence. A *transition* is a point mutation between purines (`A`, `G`) or between pyrimidines (`C`, `T`). Conversely, a *transversion* is a point mutation from a purine to a pyrimidine or vice versa. Transitions occur more often than transversions in most biological sequences [34]. If a point mutation is present in multiple organisms in a species population, it is also called a *single-nucleotide polymorphism* (SNP).

Generally, somatic mutations are differentiated from germline mutations. The former occur in somatic tissue and are not passed on to the sexual offspring, while the latter occur in reproductive cells and are passed on to the offspring [35]. Genetic variability is affected by the process of *evolution*, resulting in biodiversity. Natural selection is one of the major evolutionary processes. It describes the effect that genetic changes with phenotypes that grant a higher chance of reproduction for the organism become more common in a population. In contrast, genetic changes that impede the chances of reproduction of an organism become less common. If two organisms share a common feature that descends from the same origin, that feature is called a *homology*. Analogously, if DNA segments in different organisms descended from the same DNA segment through evolutionary processes, they are called *homologous sequences*. Homologies in the DNA are further classified into *ortholog* and *paralog* sequences depending on the evolutionary event that caused the homology. Ortholog sequences are the result of speciation events: the same DNA sequence evolved differently in groups of organisms that diverged into different species. Paralog sequences originate from a gene duplication event where a DNA fragment is duplicated and both copies of the original fragment evolve independently.

The study of evolutionary relationships between organisms is known as *phylogenetics*. The relationship between species and their evolutionary history is typically analyzed in *phylogenetic trees*. However, the use of phylogenetic trees assumes that transfer of genetic information only occurred from parent to offspring and represents the species relations accordingly: A phylogenetic tree has a *root* that represents the last common ancestor of all species within the tree. Each bifurcation from the root towards the tips of the tree represents a speciation event and constitutes the last common ancestor of all species in the subtree below. This representation is simplified and does not account for processes where DNA is passed by other means than from parent to offspring. For example, in *hybrid speciation* offspring is produced from two different species, a common incident in plants [36]. Hybridization cannot be represented by phylogenetic trees, and the same applies for horizontal gene transfers. Thus, evolutionary relationships of prokaryotes with many such transfers are often represented in *phylogenetic networks* instead.

When networks are used to study phylogenetic relationships, it is also called *reticulate evolution*. The most common approach to calculate phylogenetic trees or networks is by means of *multiple sequence alignments*. An alignment relates homologous nucleotides of the involved sequences to one another and enables the estimation of sequence divergence times based on given evolutionary models. Similarly, sequence alignments allow the calculation of the 'most likely' phylogenetic tree with standard Bayesian statistics.

2.2 Definitions

The subsequent notations are used to denote important concepts throughout this work: The set of numbers $\{1, 2, 3, \ldots, n\}$ is denoted as $[n]$. An *alphabet* Σ is a set of distinct elements, called *characters* or *symbols*. Here, it is usually assumed that Σ is the alphabet of nucleotides $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Other common alphabets in bioinformatics include the slightly modified alphabet for RNA sequences $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$, or the set of 20 distinct amino acids. It is also assumed that the elements of Σ are *ordered*, meaning that the relations $<$ and $>$ are defined for the symbols in Σ. The size of the alphabet is denoted as $|\Sigma| = o$. Each character $\sigma \in \Sigma$ is assigned its index $i \in [o]$ of the ordered list of all characters. Hence, σ_i refers to the i-th character of the ordered alphabet. Thus, for any two indices $i, j \in [o]$ with $i < j$ the relation $\sigma_i < \sigma_j$ holds. A *sequence* S is defined *over* an alphabet Σ and is a finite and ordered list of elements of Σ. A sequence is also called a *string* or a *word*. The number of elements $|S|$ in a sequence is called the *length* of S and is denoted as $n = |S|$. A sequence with $|S| = k$ is also called a *k-mer*. The characters in S are numbered from 1 to n and the i-th character is denoted by $S[i]$. A *substring* of S from the i-th character to the j-th character where $i < j$ is the string of characters $S[i], S[i+1], \ldots, S[j]$ and denoted as $S[i..j]$. A substring $S[i..j]$ where $i = 1$ is called a *prefix* of S. A substring $S[i..j]$ where $j = n$ is called a *suffix* of S. In contrast to substrings, a *subsequence* of S is a non-contiguous sequence of S. Thus, a subsequence of length k is defined by k indices $i_1, \ldots, i_k$ where $i_1 < i_2 < \cdots < i_k$; the subsequence is the concatenation of the symbols $S[i_1], S[i_2], \ldots, S[i_k]$. If there are m sequences, they are numbered from 1 to m and denoted as $S_1, \ldots, S_m$. Two strings S_i and S_j can be compared in the lexicographic order induced by Σ. For strings of different lengths where one string is a prefix of the other, the shorter string is considered to be smaller. The *Hamming* distance between two strings S_1 and S_2 of the same length n is the number of symbols at corresponding sequence positions that differ:

$$\mathrm{Ham}(S_1, S_2) = |\{S_1[i] \neq S_2[i],\, i \in [n]\}| \; . \tag{2.1}$$

For identical sequences, $\mathrm{Ham}(S_1, S_2) = 0$ applies, while $\mathrm{Ham}(S_1, S_2) = n$ holds when the sequences do not have identical nucleotides at any given sequence position. The *Edit* distance between two sequences S_1 and S_2 with lengths n_1 and n_2, respectively, is defined as the minimal number of operations required to transform one sequence into the other, whereby allowed operations include the insertion of a symbol into a sequence, the deletion of a symbol from a sequence, or the substitution of a symbol in a sequence by another symbol.

2.3 Alignment-Based Sequence Comparison

Sequence alignments are one approach to compare two or more DNA or protein sequences with respect to their evolutionary history. For this, alignments identify and relate homologous regions between the sequences to one another. *Pairwise sequence alignments* (PSAs) involve exactly two sequences and are distinguished from *multiple sequence alignments* (MSAs) which align more than two sequences. A PSA can be conceived of as a matrix with two rows, one for each sequence, and multiple columns. For DNA sequences, nucleotides of both sequences reside in the same column if the corresponding nucleotides are presumed to be homologous. Between homologous nucleotides, *gaps* are added to either of the two sequences to account for evolutionary events such as insertions or deletions (*indels*). Gaps are marked with a hyphen character at the according position in the alignment matrix; thus, sequence alignments for sequences over an alphabet Σ are constructed over an extended alphabet $\Sigma' = \Sigma \cup \{-\}$. Likewise, the concept of PSAs is applicable to multiple sequences, resulting in a MSA. A MSA for m sequences has m rows, one for every sequence; hence, a PSA is a MSA for $m = 2$. A MSA for m sequences $S_1, \ldots, S_m$ is represented as a matrix $\mathcal{A} \in \Sigma'^{m \times n}$ with m rows and n columns with $n \geq \max_i |S_i|$, see Fig. 2.3 for an example.

Another differentiation is often made between *global* and *local* sequence alignments: Global alignments assume an underlying homology that spans the entirety of the sequences. This is primarily the case if only site-level events have happened since the sequence divergence such as character substitutions or indels. Conversely, if biological events occurred that changed the whole sequence succession, for example, gene duplication or loss, translocations, inversions, or horizontal gene transfers, the sequences cannot be globally aligned in any meaningful way. Hence, global alignments are sensible when the sequences at hand are homologous in their entirety [37]; otherwise, only local alignments should be produced.

Sequence alignments were first used in 1963 to compare homologous amino acid sequences [38]. Soon after, the development of new technologies resulted in a rapid increase in the availability of DNA and RNA sequences. To properly analyze these sequences, alignments were also applied to DNA and RNA sequences [39, 40]. The number of generated sequences quickly surpassed the threshold where it was feasible to generate alignments manually, and thus, first automated programs for sequence alignments emerged. The well-known *Needleman–Wunsch* algorithm for the alignment of biological sequences was published in 1970 [41]. Hence, sequence alignments were one of the first approaches to methodologically compare the similarity of sequences and are still one of the most important tools in modern bioinformatics.

2.3.1 Models of Sequence Evolution

In order to judge the quality of a generated sequence alignment, an optimality criterion must be defined. Only then is it possible to search for the *optimal* alignment with respect to the defined criterion. Any optimality criterion inherently depends on an underlying model of sequence evolution that is assumed to have produced the sequences at hand. Optimality criteria are also referred to as *scoring schemes* and usually consist of two parts: a *scoring matrix* and *gap penalties*. A scoring matrix S with o rows and o columns defines a score S_{ij} for each pair of nucleotides $(\sigma_i, \sigma_j) \in \Sigma^2$. For a given scoring matrix S and a defined gap penalty, a *score* is assigned to an alignment by adding the respective scores and gap penalties across all columns. The resulting total score represents the quality of the alignment. The sequence alignment with the highest score out of all possible sequence alignments is considered the optimal one; the score of this 'best' alignment is also a measure for the similarity of the involved sequences. For nucleotides, such scoring schemes might be as simple as defining a

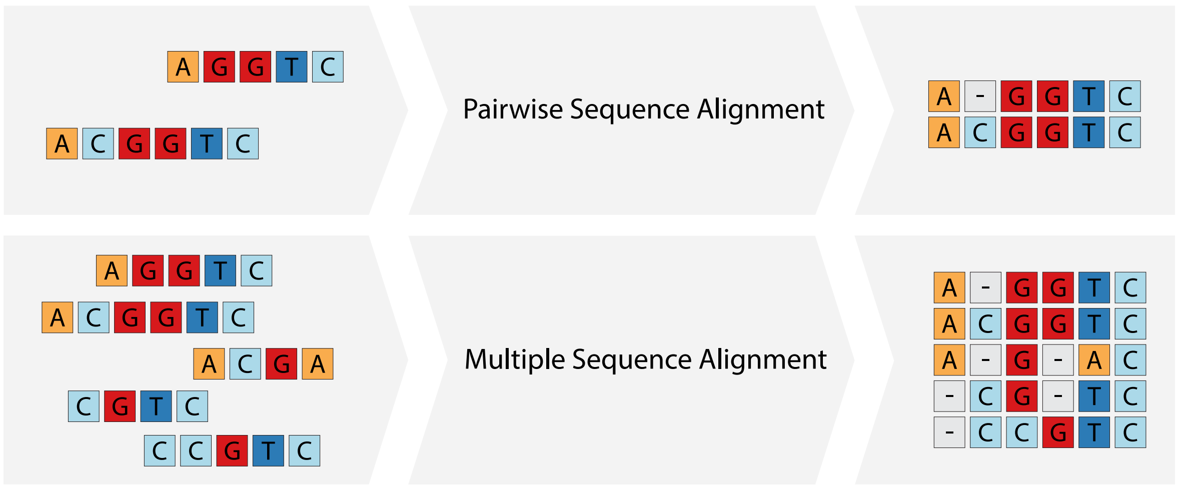


Figure 2.3 – A global pairwise sequence alignment (PSA) and a global multiple sequence alignment (MSA) between DNA sequences. The input DNA sequences (left) are allowed to have differing lengths. The alignment process adds gap characters to the input sequences in order to compute alignments (right). Each column represents those sequence positions that are assumed to be homologous. Indels are marked with a gap character in the corresponding other sequences. Both alignments were created with a freely accessible web service [42].

positive score s for identical nucleotides, and the negative of s for different nucleotides:

$$S_{ij} = \begin{cases} s, & \text{if } i = j \\ -s, & \text{if } i \neq j \end{cases} \tag{2.2}$$

However, the simple scoring scheme illustrated in Eq. 2.2 has little connection to *observed* substitution frequencies as it is not inferred from real sequence data.

To derive scoring matrices from real-world data, a *substitution matrix* M is used. Similarly to S, M is a $o \times o$ matrix where $o = |\Sigma|$ and each entry M_{ij} defines a *substitution score* between the pair of symbols $(\sigma_i, \sigma_j) \in \Sigma^2$. A substitution score M_{ij} represents the likeliness that a substitution from symbol σ_i to symbol σ_j happened. Substitution matrices are often chosen to be symmetric with $M_{ij} = M_{ji}$. Thus, it is implied that the direction of the substitution does not matter. Furthermore, a frequency vector π defines the base frequencies of the symbols in Σ. There are different methods to derive a scoring matrix from M, but often the *log-odds* approach

$$S_{ij} = \left(\frac{1}{\lambda}\right) \log \left(\frac{M_{ij}}{\pi_i \cdot \pi_j}\right) \tag{2.3}$$

is used [43], whereby λ is a scaling factor and π_i is the frequency of symbol σ_i.

To model the evolution of DNA or protein sequences over a time period, the model is usually extended to be dependent on the time t and the resulting models are referred to as *continuous-time Markov chains*. A Markov chain is defined by a *rate matrix* Q that defines the rate at which symbols of the alphabet change between one another, see Fig. 2.4. The time-dependent *transition matrix* $M(t)$ is derived from Q by matrix exponentiation:

$$M(t) = e^{Qt} \tag{2.4}$$

Each entry $M_{ij}(t)$ for two states $(\sigma_i, \sigma_j) \in \Sigma^2$ represents the probability for the mutation from σ_i to σ_j in time t. Evolutionary models are usually considered to be *stationary* Markov

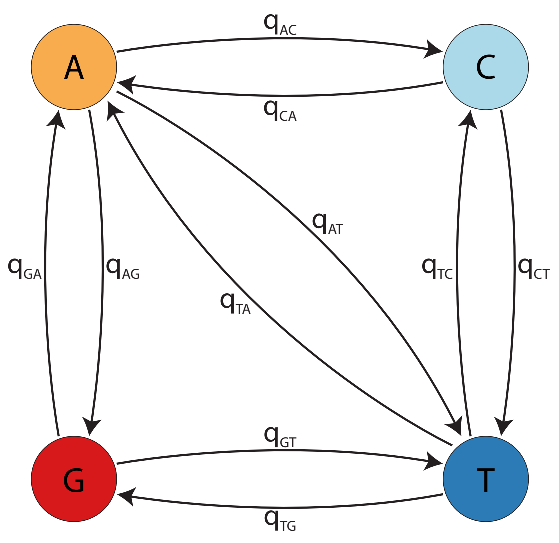


Figure 2.4 – Transition probabilities of a stationary Markov chain process for DNA evolution. The four states (round circles) correspond to the four nucleotides $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ and the transition probabilities (black arrows) between the states are marked with q_{ij}, $i, j \in \Sigma$.

processes. This means that the rate matrix Q does not change over time but stays constant. A transition matrix is called *time reversible* if the probability of a mutation between two characters is not dependent on the direction of time: $M_{ij}(t) = M_{ji}(t)$. Most continuous-time Markov chains used to model the evolution of DNA are time reversible.

Continuous-time Markov chains for DNA alignments differ with respect to the chosen rate matrix Q and the base frequency vector $\pi = (\pi_{\mathtt{A}}, \pi_{\mathtt{C}}, \pi_{\mathtt{G}}, \pi_{\mathtt{T}})$. Generally, Q and π are either inferred from the data of the current analysis at hand, or they are estimated from a large and representative data sample and reused for multiple analysis. The most general model for DNA sequences over the alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ is the *generalised time reversible* (GTR) model [44] with a total of nine free parameters. The rate matrix

$$Q = \begin{pmatrix} -(a\pi_{\mathtt{C}} + b\pi_{\mathtt{G}} + c\pi_{\mathtt{T}}) & a \cdot \pi_{\mathtt{C}} & b \cdot \pi_{\mathtt{G}} & c \cdot \pi_{\mathtt{T}} \\ a \cdot \pi_{\mathtt{A}} & -(a\pi_{\mathtt{A}} + d\pi_{\mathtt{G}} + e\pi_{\mathtt{T}}) & d \cdot \pi_{\mathtt{G}} & e \cdot \pi_{\mathtt{T}} \\ b \cdot \pi_{\mathtt{A}} & d \cdot \pi_{\mathtt{C}} & -(b\pi_{\mathtt{A}} + d\pi_{\mathtt{C}} + f\pi_{\mathtt{T}}) & f \cdot \pi_{\mathtt{T}} \\ c \cdot \pi_{\mathtt{A}} & e \cdot \pi_{\mathtt{C}} & f \cdot \pi_{\mathtt{G}} & -(c\pi_{\mathtt{A}} + e\pi_{\mathtt{C}} + f\pi_{\mathtt{G}}) \end{pmatrix} \tag{2.5}$$

has six free parameters (a to f) that define the rate changes between nucleotides. The lower left and upper right of Q are dependent on each other to satisfy the time-reversibility. The base frequency vector sums up to 1 and thus has three free parameters. Several simpler models with a reduced number of free parameters exist: The *Jukes-Cantor* model [45] is the simplest and assumes equal base frequencies as well as equal substitution probabilities between all nucleotides. The K80 model [46] distinguishes between transitions and transversions. Thus, K80 reflects the difference in mutation frequency in the model by assigning lower mutation rates for transversion events than for transition events; the base frequencies are set to equal. There are many additional models for the evolution of DNA that allow a subset of the parameters of the GTR model [47–51]. Amino acids vary in their similarity based on their structure and chemical properties. This results in strongly differing substitution probabilities [52]. Thus, estimating accurate substitution models for amino acid sequences is more complex. Substitution matrices have been inferred from the relative substitution events observed in reference data sets on several occasions: Common models include *position accepted mutation* matrices [53], and many variants thereof [54]. Another set of often used matrices are *block substitution* matrices [43]. Many iterations of updated substitution matrices for amino acid sequences emerged, such as improved versions of PAM [54], matrices based on larger datasets [55], or matrices estimated with other methods such as the maximum likelihood principle [56].

For both nucleotides and amino acids, alignment gaps are penalized with a negative score. With a *linear* gap penalty, each gap character receives the identical negative score. An *affine* gap penalty differentiates between the new opening of a gap and the extension of an existing gap. It is common practice to consider all nucleotide sites in the alignment to be independent from each other. The *total score* of a PSA with respect to a scoring scheme is then the sum over all scores of each nucleotide column in the alignment plus the gap penalties. For multiple sequence alignments, the total score is often calculated with the *Sum-of-Pairs* approach: Here, the scores of all possible pairs of sequences in the MSA are calculated first, and all pairwise scores are summed up subsequently. It is also possible to account for *rate heterogeneity* [57]; then, the parameterization of the substitution model and base frequencies is adjusted for different columns of the alignment.

2.3.2 Calculating Sequence Alignments

Calculating pairwise and multiple sequence alignments is computationally expensive with respect to time and memory requirements. For two sequences of length n there are already $(2n)!/n!^2$ possible sequence alignments [58]. Finding the optimal PSA with respect to a scoring criterion takes time in the order of $\mathcal{O}(n^2)$. Calculating an optimal MSA is NP-hard [59] and, therefore, there is no feasible algorithm to compute the optimal MSA for any substantial number of sequences. Instead, algorithms calculate MSAs by using approximate heuristics to drastically speed up the computations, at the cost of suboptimal solutions. One of the first algorithms for pairwise sequence alignments, the Needleman-Wunsch algorithm [41], calculates the optimal *global* alignment between two sequences using *dynamic programming*: In the first step, the algorithm iteratively determines optimal alignments for prefixes of increasing length and creates a table of all respective alignment scores. In the second step, a traceback through the obtained scoring table of all prefix alignments yields the optimal global alignment. For two sequences of lengths n_1 and n_2, the Needleman-Wunsch algorithm runs in $\mathcal{O}(n_1 \cdot n_2 / \log n_1)$ time complexity and $\mathcal{O}(n_1 \cdot n_2)$ space complexity [60]. In contrast, the Smith-Waterman algorithm [61] finds an optimal *local* sequence alignment between two sequences and operates similarly to the Needleman-Wunsch algorithm. And, likewise, it requires quadratic time and space with respect to the product of the lengths of both input sequences. Many variants of both algorithms exist that use modern computer architecture or novel data structures to improve upon the original versions, such as GPU-accelerated versions [62, 63], a version that requires only linear space [64], or a version that brings further speed improvements [65].

As sequence alignments are one of the major approaches to quantify sequence similarity in an evolutionary context, they play an essential role in diverse use cases: Determining DNA or protein homology, screening unknown sequences against existing databases for sequence identification, SNP analysis [66], genome assembly, building phylogenetic trees, and many more [67], see Subsec. 2.3.3. However, it is often not imperative, and due to the time complexity not advisable, to calculate an optimal MSA. Instead, even when multiple sequences are involved, using PSAs between a subset of sequence pairs or using suboptimal MSAs inferred with adequate heuristics is often sufficient to answer common research questions. But even calculating optimal PSAs between many pairs of sequences is too time-intensive for most applications. Thus, several tools exist that compute approximate pairwise sequence alignments or derived similarity measures thereof. One of the most influential tools for sequence comparison is the *basic local alignment search tool*, or short BLAST [68]. BLAST is designed to efficiently align a single *query sequence* to multiple other sequences within a database by employing fast heuristics. The output of BLAST is a collection of high-scoring local alignments between the query sequence and database entries. However, BLAST does not guarantee to find the overall optimal local alignment for a given query. The general idea is to rapidly detect candidate regions in the database sequences that have a high probability to form high-scoring alignments with the query. This approach is also known as *seeding*. In BLAST, the seeding approach works as follows: The query sequence S_q over the alphabet Σ is divided into words w_k of a specified length k that form a set $K_{S_q} = \{w_k,\, w_k \text{ is substring of } S_q\}$. Then, a word $w'_k \in \Sigma^k$ is called a *neighboring word* to a word $w_k \in K_{S_q}$ with respect to a substitution matrix M when the score between w'_k and w_k is above a defined threshold t. Here, the score between w_k and w'_k is calculated as if the two words would constitute a sequence alignment without gaps. BLAST determines all neighboring words w'_k to any word $w_k \in K_{S_q}$ and stores them in a set. Thus, the resulting set of neighboring words represents all words that are identical or exhibit at least a certain similarity to any word of length k in S_q. The reference database is then searched for all neighboring words of S_q and the according sequences are extracted; these exact word matches represent the initial seeds for local alignments. Each seed in each reference sequence is extended in both directions to form a local sequence alignment based on the substitution matrix M. Extension of alignments is stopped when the score decreases over multiple consecutive nucleotides. All resulting alignments between the query and any database sequence that exceed a predefined similarity threshold are kept as the output of BLAST. By now, a whole family of different BLAST algorithms for a multitude of applications exists: Gapped BLAST, PSI-BLAST [69], and BLAT [70] are faster versions with improved heuristics for the seeding and alignment extension, PLAST [71] is a parallelised version, Mega-BLAST [72] performs additional preprocessing of the database to speed up the database search, and Magic-BLAST [73] is designed to handle RNA data. There is a wealth of other sequence similarity search tools such as FASTA [74], UBLAST and USEARCH [75], or PatternHunter [76]. All of these programs are predominantly designed to search large reference databases for entries that are similar to a supplied query sequence and likely to produce high-scoring pairwise alignments.

There are also many tools available for the calculation of multiple sequence alignments. Due to the large time and space requirements when constructing MSAs, present sequence aligners use ever-improving heuristics to compute near-optimal alignments in a fast manner: The most common approach is the *progressive* construction of alignments [67]. Progressive alignment algorithms iteratively combine pairwise alignments, from the most similar sequences to the least similar ones [77]. Often, a *guide tree* is used that specifies the order in which PSAs are combined to arrive at the final MSA. The guide tree is determined from a fast initial pairwise sequence similarity comparison. In some algorithms pairwise alignments are fixed, while in other implementations pairwise alignments can be adapted later on when incorporating new sequences. One of the first popular software tools that pursued this paradigm was ClustalW [78], as well as its extension ClustalX [79] which provides a graphical user interface. Compared to ClustalW, the program DIALIGN [80] and its successor DIALIGN-T [81] follow a *segment-based* approach to create DNA and protein MSAs. Here, the MSA is based on pairwise sequence similarities of local scope; based on such highly similar substrings between sequence pairs, it is possible to combine distantly related sequences in a single MSA. The resulting alignments are of high quality, especially for locally related sequences. MAFFT [82] is a popular progressive sequence alignment method that relies on the Fast Fourier Transform to quickly determine homologous regions between input sequences. In general, progressive alignment methods are susceptible to differences in the guide tree and a bad guide tree may cause sub-par results [83]. Thus, many progressive alignment programs, such as MSAProbs [84], employ a step referred to as *iterative refinement* to overcome such limitations. Subsequent versions of MAFFT also include iterative refinements of the produced sequence alignments [85], as well as parallelization [86], and a multitude of further enhancements [87, 88]. Other alignment tools that follow a progressive procedure are, for example, T-Coffee [89] and MUSCLE [90]. *Hidden Markov models* (HMMs) are another strategy for the construction of multiple sequence alignments [91]. In HMMs, alignments are represented as directed acyclic graphs where sequence characters are represented as nodes, also called *observed states*. The *hidden states* are characters of the mutual ancestral sequence of all aligned sequences. HMMs are not only used for the construction of MSAs, but also with respect to a variety of other tasks in biological sequence analysis, such as the classification of sequences, annotation of genes, or database similarity search [92]. Normally, *profile HMMs* are employed which encode not only nucleotide frequencies but also insertions and deletions, all of which are encoded position specific [93, 94]. Hence, profile HMMs are probabilistic models capable of comprehensively representing whole sequence sets or gene families when parameterized accordingly. SAM [91] was one of the first approaches to use HMMs for building MSAs. HMMER [95] is a popular tool for sequence alignments and database search based on HMMs. Clustal Omega [96] uses a combined approach of progressive sequence alignments and HMMs based on the HH-suite [97]. It is also possible to compare profile HMMs with each other; this enables the detection of even more distant relationships between sequences and is, for example, utilized by the program HHsearch [98]. Another group of MSA algorithms are *phylogeny-aware* sequence aligners. The main distinguishing factor of phylogeny-aware approaches is their focus on the construction of alignments that also integrate the phylogenetic context of the sequences into the estimation procedure. As a consequence, a resulting alignment is not necessarily the one that maximizes the similarity between the sequences [67]. Currently, various software packages are available that perform phylogeny-aware multiple sequence alignment [99–101] and several comprehensive reviews on multiple sequence alignment tools in general exist [67, 102, 103].

2.3.3 Applications of Sequence Alignments

The applications of sequence alignments are broad and varied. Alignments have contributed substantially to bioinformatic data analysis since they were first devised: Mostly, they serve as a means to quantify similarity between DNA, RNA, or protein sequences. The alignment score of a PSA specifies the similarity of the two involved sequences, but several other measures, such as the Hamming or Edit distance, are also derived from DNA alignments. In addition, there are a variety of models that estimate the number of nucleotide substitutions from alignments [48, 104–107], which in turn is utilized to estimate the divergence time of the according species [108, 109]. Furthermore, identification and classification of newly obtained sequences is often one of the first steps in analyses pipelines, and programs such as BLAST or PatternHunter [76] employ alignment strategies based on the seed-and-extent approach to scan biological databases for known homologs. Homologous sequences then provide information about taxonomic affiliation of the new sequences, but they are also utilized to transfer other known information, for example, about secondary or tertiary structure for proteins. In this context, alignments have been used for protein structure prediction [110, 111], phenotype prediction [112], or functional prediction [113, 114]. Furthermore, alignments are used for motif finding [115, 116] and gene prediction [117]. There are also many sequence aligners optimized for specific sequence types, such as short metagenomic reads [118], whole genomes [119], or splice-aware aligners [120]. Another major use case of MSAs is the reconstruction of phylogenetic trees. For example, maximum-likelihood models for phylogeny reconstruction infer the phylogenetic tree with the highest log-likelihood for a given alignment and a large variety of tools exist for this task [121–123], see Subsec. 2.5.4. Some of these approaches estimate alignments and phylogenetic trees simultaneously [101, 124]. Alignments are also used for *reference-based* sequence assembly: Here, short reads are aligned against reference genomes to guide the assembly process [125, 126]. For this purpose, several tools are specialized for aligning short query reads against large reference genomes [127, 128].

Evidently, pairwise and multiple sequence alignments are an important tool in many areas of bioinformatics and form the basis for many computational analyses. Nonetheless, alignments also come with inherent limitations and drawbacks. In general, the calculation of MSAs is time-intensive even when multiple heuristics are applied in an attempt to quickly discover local optima in the large space of possible alignments. Thereby, depending on the software in use, the heuristics used to speed up the calculation can result in sup-optimal alignments. This distorts subsequent analysis steps and, for example, might result in inaccurate phylogenetic trees. Additionally, sequence alignments are best applicable if the sequences share global homologies. However, biological mechanisms such as gene duplication or loss, horizontal gene transfers, high mutation rates, or genome rearrangements result in sequences that are difficult to compare with traditional global alignment-based approaches. Many tools have been developed to specifically deal with local alignments [129–131] or whole genome alignments [132–134]. Still, it remains questionable whether alignments constitute the best approach when considering such data sets [135] and it has been suggested that the quality of whole genome alignments varies considerably [136]. With the rise of next-generation and third-generation sequencing techniques, the amount of genomic data is increasing at higher speeds than ever before. Modern sequencing data often consist of short sequencing reads or draft genomes, where sequences are not necessarily assembled to longer contiguous sequences; see Section 2.6. This also fundamentally limits the use of alignments in these contexts. The data has also reached a volume where MSA-algorithms are not sufficiently fast, even when broad heuristics are used, exacerbating the problem of long runtimes. Additionally, alignments are always based on assumptions such as the underlying evolutionary model, substitution matrices, gap penalties, and others. Using adequate assumptions for the data at hand is necessary to infer well-founded alignments, and any misspecification can result in undesirable alignment results. For all of these reasons, another group of methods for the calculation of sequence similarity emerged that do not rely on sequence alignments: so-called *alignment-free* sequence comparison methods.

2.4 Alignment-Free Sequence Comparison

Algorithms for alignment-free sequence comparison are a broad category of methods that do not rely on the calculation of pairwise or multiple sequence alignments to determine the similarity between biological sequences. Instead, other properties of the sequences are used to derive measures of sequence similarity. Most approaches can be assigned to one of two groups: *word-based* methods or *information theory-based* methods, although there are approaches that fit in neither of these categories [137]. In word-based methods, sequences are transformed into a new *feature space* with the reasoning that it is computationally less expensive to compare sequence representations in the newly constructed space than to compare the original sequences themselves. Common feature spaces are based on the distribution or abundance of substrings of the sequences. Such feature representations are often *position-independent*: Similar sequence fragments from two sequences are detected, even if they occur at different positions within the original sequences. Therefore, these methods are typically less influenced by large-scale evolutionary mechanisms, such as genome rearrangements. This property gives word-based methods a unique advantage in comparison to alignment-based algorithms. Information theory-based methods use concepts from the field of information theory, such as *entropy* of or *mutual information* between sequences, to quantify sequence similarity [138]. On the one hand, alignment-free methods tend to be faster than alignment-based methods and, thus, are commonly used when dealing with large data sets. On the other hand, alignment-free methods are generally assessed to be less accurate than alignment-based methods [139].

2.4.1 Information Theory-Based Methods

The *Shannon entropy* is a measure of uncertainty for the possible outcomes of a random variable X. A high entropy implies that the outcome of a single draw of X is highly uncertain; or, in other words, that previous draws from X do provide little *information* about future draws. A biological sequence can be interpreted as a series of draws from a random variable where the values are drawn from the according alphabet Σ, such as $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ for DNA sequences. Sequences with many repetitions have a low entropy, whereas more complex sequences have a high entropy. The notion of *compression* is closely related to entropy: Sequences with low entropy are highly compressible without loss of information, while little compression can be applied to sequences with high entropy. It is possible to apply the concepts of entropy and compression in order to infer the degree of similarity between two sequences. Let S_1 and S_2 be two sequences that are already fully compressed, meaning that no further lossless compression is possible. The concatenation of the sequences is denoted as S_{1+2}. The amount of compression that can be applied to S_{1+2} contains information about the similarity of S_1 and S_2: If S_1 is identical to S_2, then S_{1+2} has a low entropy and S_{1+2} is easily compressible while retaining all information. However, when S_1 and S_2 do not share common information, S_{1+2} is impossible to compress without loosing information. Thus, the degree of compression that can be applied to S_{1+2} is a measure of similarity for S_1 and S_2.

A large number of compression strategies for biological sequences or whole sequence databases are available [140], as are derived distance metrics and applications [141, 142]. In this context, compression has also been used to discover large-scale evolutionary events such as genome rearrangements [143]. Several other algorithms use metrics directly related to sequence entropy that yield information about the similarity of two sequences. For example, the *Kullback-Leibler divergence* [144] represents the amount of information that is shared between two sequences. Another related metric is the *Jensen-Shannon divergence*, calculated, for example, by the software ALFREE [137]. Interestingly, it is possible to estimate such metrics from word-frequency statistics as well: One method following this approach is the AVERAGE COMMON SUBSTRING method [145]. It defines a similarity measure between two sequences S_1 and S_2 with lengths n_1 and n_2 based on the average length of maximum common substrings. For every position $i \in [n_1]$ in S_1 the longest matching substring in S_2 starting at some arbitrary position $j \in [n_2]$ is determined. The resulting lengths $l(i)$ are averaged over all positions $i \in [n_1]$ to obtain the average length of the common substrings $\text{acs}(S_1, S_2) = \sum_{i\in[n_1]} l(i)/n$. To obtain a distance that is symmetrical and also satisfies $d(S_1, S_1) = 0$ the inverse is computed and then normalized:

$$d'(S_1, S_2) = \frac{\log(n_2)}{\text{acs}(S_1, S_2)} - \frac{\log(n_1)}{\text{acs}(S_1, S_1)}\,, \tag{2.6}$$

$$d(S_1, S_2) = d(S_2, S_1) = \frac{d'(S_1, S_2) + d'(S_2, S_1)}{2}\,. \tag{2.7}$$

The authors demonstrate that this measure is directly related to the Kullback-Leibler relative entropy. The implementation runs in $\mathcal{O}(n_1)$ by using suffix trees [146], a large gain in computation speed compared to alignment-based approaches.

2.4.2 Word-Based Methods

Word-based methods use the occurrence, distribution, or other properties of substrings or subsequences present in the input sequences. They are rarely concerned about the overall order and identity of all nucleotides in the sequences but instead focus on the presence and composition of shorter sequence parts, also called *words* or *k-mers*. The simplest implementation of this idea is to compare the number of occurrences of k-mers for a specified length k between the provided sequences. The reasoning is that closely related sequences share many k-mers since only a limited number of mutations occurred since their divergence; in contrast, distantly related sequences share far fewer identical k-mers due to an increased number of mutations and indels that occurred. Assuming that there are two sequences S_1 and S_2 of length n and that all positions mutate independently with a probability p over time t, then the probability to observe the same k-mer at any position i in both sequences is $(1-p)^{k^t}$, provided that no indels or other evolutionary events occurred. The overall number of expected identical k-mers at all corresponding positions is accordingly given as $n \cdot (1-p)^{k^t}$. However, k-mers are usually not only compared at identical sequence positions, but all occurrences of k-mers across all sequence positions are taken into account. In this way, k-mers from homologous sequence regions that reside at different absolute locations in S_1 and S_2 are detected as well. Still, it is extremely unlikely to observe an identical k-mer at non-homologous positions for large values of k: For DNA sequences over an alphabet of four symbols, under the simplified assumption of independent positions with random nucleotides, this probability is 0.25^k; already for $k = 12$ this evaluates to less than 10^{-7}. This is an important aspect of word-based methods, as it is safe to assume that the presence of identical k-mers stems from an evolutionary interrelation and provides valuable information. Even when identical k-mers are found at different positions in the sequences S_1 and S_2 they most likely do not appear by pure chance. This example also highlights an important advantage of word-based methods over alignment-based methods: Most word-based methods are unaffected by large-scale evolutionary events, such as genome rearrangements, as they do not take overall sequence order into account.

Given the previous considerations, one of the most basic algorithms for comparing a set of m sequences $S_1, \ldots, S_m$ of possibly differing lengths is as follows: For a specified size k, the occurrences for all distinct k-mers in a sequence S_i, $i \in [m]$ are counted. For an alphabet

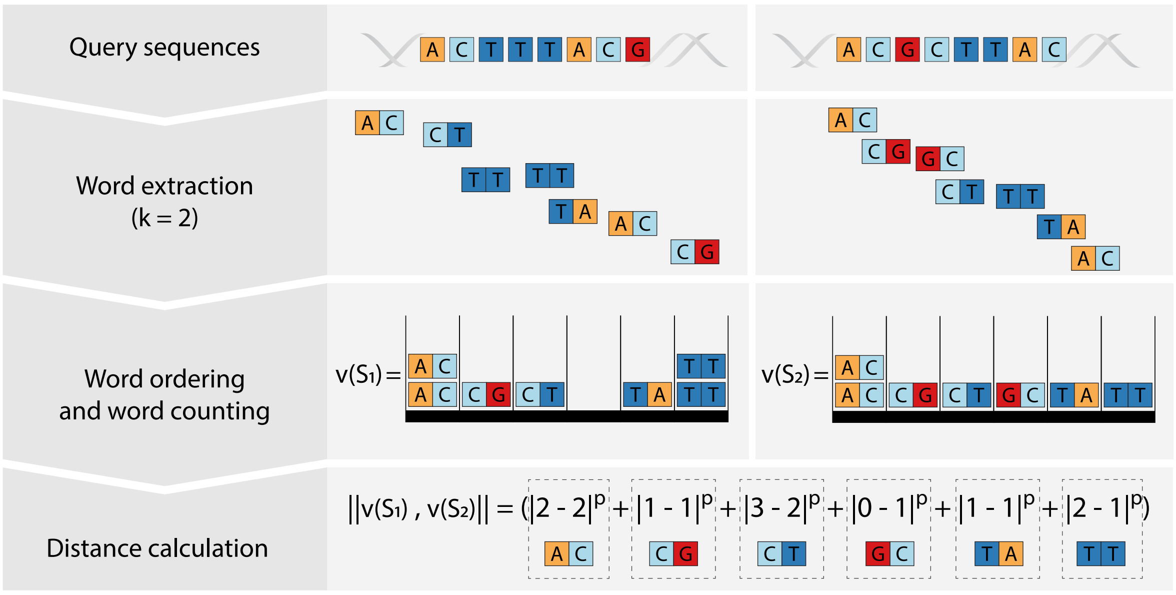


Figure 2.5 – A simple word-based procedure for sequence comparison: Two query sequences are decomposed into consecutive words of length k, here $k = 2$. Then the number of each unique k-mer in S_1 and S_2 is counted, resulting in a word-count array $\mathbf{v}(S)$ for each sequence. Those k-mers that are not present in either of the species are omitted to save memory space. The distance between S_1 and S_2 is calculated on the basis of the the two word-count arrays $\mathbf{v}(S_1)$ and $\mathbf{v}(S_2)$, for example, a Minkowski distance.

Σ, the number of possible words of length k is $l = |\Sigma|^k$. Since an ordering of the k-mers $K_1, K_2, \ldots K_l$ is induced by the order of Σ, the resulting values for a sequence S_i can be stored in a vector $\mathbf{v}(S_i)$ of size l where the j-th element $v_j(S_i)$ corresponds to the count of K_j. Thus, each entry $v_j(S_i)$ in the vector corresponds to the number of occurrences of K_j in S_i according to their lexicographic order. Extracting the k-mers of a sequence S_i can be carried out efficiently with a sliding window of length k that is shifted from the first position $S_i[1]$ of the sequence to the position $S_i[n-k+1]$. At every position $\iota \in [n]$ the count of the k-mer $S_i[\iota : \iota + k]$ is updated at its according position in the vector $\mathbf{v}(S_i)$. Subsequently, such vectors are created for all other sequences as well. Then, two sequences S_{i_1} and S_{i_2} can be compared with respect to their lists of k-mer counts $\mathbf{v}(S_{i_1})$ and $\mathbf{v}(S_{i_2})$. Here, any metric for the comparison between two vectors can be used, for example, *Minkowski-metrics*

$$\|\mathbf{v}(S_{i_1}), \mathbf{v}(S_{i_2})\|_p = \left(\sum_{j=1}^{l} |v_j(S_{i_1}) - v_j(S_{i_2})|^p \right)^{\frac{1}{p}} . \tag{2.8}$$

For $p = 1$ this equals the so-called *manhatten* distance, for $p = 2$ the *euclidean* distance, and for $p \mapsto \infty$ the *supremum* distance. This procedure is also illustrated in Fig. 2.5 and comprehensive overviews of possible distance functions for k-mer counts of biological sequences and their advantages and drawbacks exist [147]. However, this method also comes with computational difficulties: For a given k and alphabet Σ the number $l = |\Sigma|^k$ of possible words of length k grows rapidly. For the nucleotide alphabet and $k = 31$, a value used in certain programs [148], storing all possible k-mers would result in $4^{31} \sim 4.611 \cdot 10^{18}$ values. Storing a vector of this size for even a single sequence is inconceivable with current technology and would require more than one billion gigabytes of space. For the substantially larger amino acid alphabet, the number grows even faster with increasing values of k. Hence, an implementation that stores the counts for all possible k-mers is not feasible. Instead, only those k-mers that are present in at least one sequence under consideration are stored; often singular k-mers are omitted from being stored as well and instead they are dismissed as 'spurious' k-mers. In general, the choice of k is a key factor in all word-based approaches to sequence comparison: Choosing small values for k results in an insufficient characterization of the sequences and k-mer vectors of different sequences converge to similar distributions. On the contrary, large values of k result in only a few or no common k-mers across different sequences, which prohibits meaningful comparison. These opposing trends always result in a trade-off when choosing appropriate values of k.

A closely related approach to compute similarities between DNA sequences is the use of *min hash techniques*. Here, the main insight is that it is not necessary to save all k-mers of the sequences to estimate similarities. Instead, a subset of all k-mers, a so-called *sketch*, is sufficient. In this way, memory requirements and runtimes are significantly reduced. The tool Mash [149] is one technique that adopts this principle: Given two sequences S_1 and S_2 and their k-mer sets $W_{S_1} = \{S_1[\iota : \iota{+}k],\ \iota \in [n_1 - k + 1]\}$ and $W_{S_2} = \{S_2[\iota : \iota{+}k],\ \iota \in [n_2 - k + 1]\}$, Mash calculates the similarity between the sequences by estimating the Jaccard index.

$$J(W_{S_1}, W_{S_2}) = \frac{|W_{S_1} \cup W_{S_2}|}{|W_{S_1} \cap W_{S_2}|}\,. \tag{2.9}$$

The Jaccard index is then used to infer the Hamming distance. However, instead of considering the complete k-mer sets, *Mash* passes every k-mer through a universal hash function h resulting in the set $W_h = \{h(K_j),\ j \in [l]\}$. For a sequence S it then selects only those k-mers K_j from W_S where $h(K_j)$ is among the lowest s values of W_h, whereas s is the predefined *sketch size*. This results in a randomly selected subset of all possible k-mers while it is guaranteed that the same k-mers are selected across different sequences. Calculating the Jaccard index for these subsets is an unbiased estimate of the overall Jaccard index. This allows Mash to scale to very large sequence databases without running into computational troubles. Several adaptations to the approach exist, for example, Order Min Hash [150] which estimates the Edit distance instead of the Hamming distance by incorporating the relative order of k-mers into the sketches. Hashing techniques are applied in diverse applications: For example, MashMap [151] maps long reads to existing reference genomes, but hashing has also been used for genome assembly [152] or SNP detection [153].

As for hash-based techniques, there is also a large variety of other approaches that utilize other types of substrings or different algorithmic procedures to calculate sequence similarity. The shortest unique substring algorithm shustring identifies unique substrings in sequences that cannot be extended to the left or right without losing their uniqueness [154]. The distribution and number of shortest unique substrings is then used, for example, to detect repeats or singular regions in a genome. kmacs [155] is an algorithm that extends the average common substring approach, however, with an important difference: kmacs allows up to k positions in a substring to deviate between the two sequences (whereas this k is not to be confused with the word length k of other substring approaches). This approach poses a new and unique strategy for alignment-free sequence comparison as it does neither rely on continuous substrings nor on subsequences of a fixed length. Complementary to this approach, the authors also developed a method based on non-contiguous substrings of fixed length k, so-called *spaced words*. These spaced words are defined by a binary pattern, see also Subsec. 2.4.3, and their frequencies are used to infer a measure for sequence similarity. Although this idea has previously been applied to search large databases [76], kmacs applied it to different tasks such as phylogeny reconstruction. The main advantage of using non-contiguous words is that identical nucleotide positions are spread out and, thereby, are statistically less dependent on each other compared to using contiguous words. The use of such *inexact* k-mers has been demonstrated to produce better results than methods that use only exact k-mers [76]. Other alignment-free word-based approaches use minimal absent words [156, 157], perform variable-length encoding of nucleotides depending on their surrounding sequence parts [158], or use statistical metrics of irredundant common subwords [159]. Several comprehensive reviews for alignment-free methods exist [137, 160, 161].

One drawback of the methods introduced so far is their inability to compute similarity or dissimilarity measures that entail a direct biological interpretation. While, for example, the Euclidean distance between two k-mer count vectors provides information about the relative degree of relatedness between the sequences, it cannot be used to infer information about biological properties of the sequences themselves. In contrast, alignment-based approaches allow the calculation of biological properties of the underlying sequences, such as the average number of nucleotide substitutions per sequence position, parsimonious measures that describe the minimal number of evolutionary events that explain the data, or the relative number of shared genes between organisms. The alignment-free program Co-phylog [162] overcomes this limitation: It finds pairs of k-mers that occur next to each other in both sequences and are separated by exactly one base pair. They term the single nucleotide *object*, and the surrounding identical words *context*. If the same *context* is present in both sequences, it is called a *micro-alignment*. Such micro-alignments have identical nucleotides at all positions of the context, while an arbitrary nucleotide is allowed at the object. Then, the phylogenetic distance between sequences is estimated as the average number of times that the object is different across all micro-alignments. The software andi [163] extends the approach of Co-phylog by allowing arbitrarily large regions between the aligned contexts.

2.4.3 Filtered Spaced-Word Matches

Estimating the number of nucleotide substitutions with andi or Co-phylog requires that micro-alignments are exclusively formed from homologous sequence regions. Only then do the derived nucleotide substitution statistics reflect the actual evolutionary distance of the sequences. However, the general issue of word-based methods regarding the choice of k also applies here and causes a trade-off for the length of the context: On the one hand, a short context context results in a larger number of micro-alignments that occur purely by chance and such spurious micro-alignments skew distance estimates. On the other hand, a long context only works when the sequences under consideration are sufficiently similar.

The alignment-free program Filtered Spaced-Word Matches (FSWM) [164] aims to overcome this limitation. FSWM estimates the average number of nucleotide substitutions between every pair of a set of sequences using spaced words. For this, a binary pattern $P \in \{\mathtt{0}, \mathtt{1}\}^l$ of a predefined length l is used. The 1's in the pattern are termed *match* positions, while the 0's are called *don't care* positions. The number of 1's in P is called the *weight* of P. Given an alphabet Σ and an extended alphabet $\Sigma_+ = \Sigma \cup \{*\}$ where $*$ is the so-called *wildcard* character, a spaced word of length l with respect to the pattern P is defined as $W \in {\Sigma_+}^l$ where $W[\iota] \in \Sigma$ if ι is a match position of P and $W[\iota] = *$ if ι is a don't care position in P. A spaced word that starts in a sequence S at position i is called a spaced-word occurrence and written as (W, i). For example, a possible spaced word over the alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ and the pattern $P = \mathtt{110101}$ is provided by $W = \mathtt{TT*A*C}$. This spaced word has a spaced-word occurrence $(W, 2)$ in the sequence $S = \mathtt{GTTGATCA}$. Two sequences S_1 and

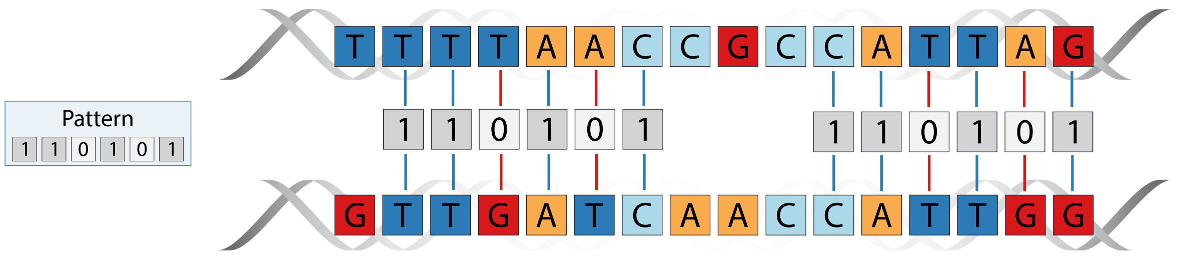

Pattern
1 1 0 1 0 1
T T T T A A C C G C C A T T A G
1 1 0 1 0 1
1 1 0 1 0 1
G T T G A T C A A C C A T T G G

Figure 2.6 – Spaced-word matches between two DNA sequences: A binary pattern $P =$ `110101` is defined. For the given input sequences, there are two spaced-word matches with respect to P. Nucleotides at the match positions are identical in both sequences (blue lines). The don't care positions have identical *or* different nucleotides (red lines).

S_2 have a spaced-word match at positions i and j, respectively, if the same spaced word W has a spaced-word occurrence in sequence S_1 at position i and sequence S_2 at position j. That means, when P is aligned to S_1 starting at position i and at S_2 starting at position j, the nucleotides at the match positions of P are identical in both sequences, while the don't care positions can be either identical or different:

$$S_1[i + \iota] = S_2[j + \iota], \text{ if } \iota \text{ is a match position of } P\,. \qquad (2.10)$$

FSWM considers patterns with a length of 112 and a weight of 12 by default. The positions of the match positions within P have little influence on the performance. Hence, spaced-word matches are similar to the earlier introduced micro-alignments, however, they are defined with respect to a more flexible binary pattern P. Nonetheless, the term *micro-alignment* is also used when referring to spaced-word matches. Figure 2.6 shows an example of spaced-word matches between two sequences.

For a substitution matrix M that defines substitution scores for all pairwise combinations of symbols in the alphabet Σ, the *score* of a spaced-word match is defined as the sum of all substitution scores between the symbols at the don't care positions of the two spaced-word occurrences, see Fig. 2.7. If the score of a spaced-word match is above a preset threshold t, the match is considered to originate from two homologous sequence regions. On the contrary, if the score is below t, the match is considered a random (or *background*) match that appeared purely by chance and not due to sequence homology — t is set to 0 as default. FSWM determines all spaced-word matches between two DNA or RNA sequences, calculates their scores and removes all matches with a score below t. The remaining matches are used to estimate the average number of nucleotide substitutions: First, the average number of substitutions at the don't care positions are counted and then the resulting percentage is adjusted with the Jukes-Cantor formula [45]. In ANDI and CO-PHYLOG the context must be sufficiently long to ensure that only homologs are considered; in contrast, the filtering procedure overcomes this requirement in FSWM. Even with a small weight of P the non-homologous random matches are filtered out based on their score. After filtering, a spaced word W can occur at multiple positions in both sequences. If W occurs at p positions in S_1 and q positions in S_2 there are $p \cdot q$ possible spaced-word matches for W. This poses two problems: First, it takes a quadratic amount of time to consider W, which is especially troubling for words W with many spaced-word occurrences. Second, considering all pairwise matches results in double counting of sequence positions as every spaced-word occurrence is matched with multiple other spaced-word occurrences. As spaced-word matches represent homologous regions, it is

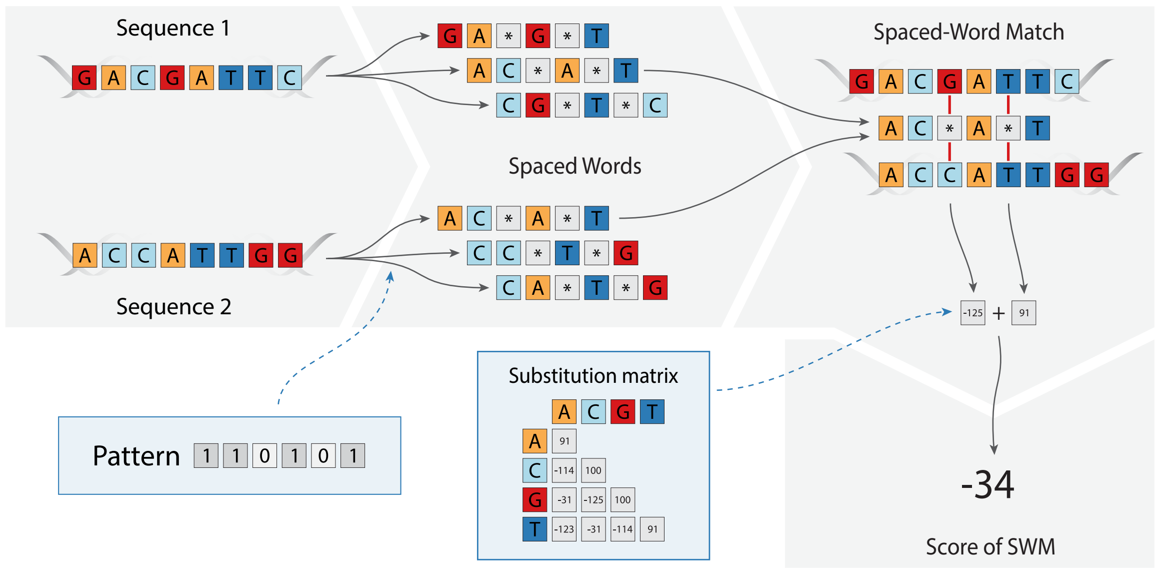


Figure 2.7 – The filtered spaced-word matches approach to calculate the number of nucleotide substitutions between two sequences: Based on a predefined pattern P, all spaced words between the two query sequences are extracted. Identical spaced words form a spaced-word match and its score is calculated on the basis of a substitution matrix M. Spaced-word matches with a score below a predefined threshold t are removed from further consideration. From all remaining spaced-word matches, the average number of nucleotide substitutions at the don't care positions is calculated and corrected with the Jukes-Cantor formula. Note that the pattern presented here only serves the purpose of visualizing how spaced-word matches are applied; real patterns are longer and mostly comprised of don't care positions.

natural to limit every spaced-word occurrence to a single match. Therefore, instead of using all pairwise matches between the occurrences of a spaced word, FSWM uses a heuristic to assign each spaced-word occurrence from one sequence to exactly one spaced-word occurrence from the second sequence. The scores of the spaced words are used in a greedy procedure: First, the occurrences that form the spaced-word match with the highest score are selected and removed from the list of occurrences. Then, the two remaining spaced-word occurrences with the highest of the $(p-1) \cdot (q-1)$ scores are removed from the list. This is repeated until no further spaced-word occurrences remain for either S_1 or S_2. This process assumes that the spaced-word occurrences of the spaced-word match with the highest score in each iteration are two homologous sequence regions. Note, however, that double counting of single sequence positions still occurs when spaced-word occurrences overlap each other. For m input sequences, FSWM calculates their pairwise distances with the described procedure and generates a $m \times m$ distance matrix $\mathbf{M}$. Phylogenetic trees can be produced from $\mathbf{M}$ with standard methods such as neighbor joining, see Subsec. 2.5.2. It was also proposed to use a set of multiple patterns $\mathcal{P}$, instead of a single pattern P; the above procedure is then performed for each pattern $P \in \mathcal{P}$ individually, and the resulting distances are averaged over all patterns. Using multiple patterns reduces the variance of distance estimates that is attributed to the choice of a single pattern. When using multiple patterns, it is recommended to optimize them with respect to their *overlap complexity*, for example, with the program RASBHARI [165].

Since the introduction of the program FSWM, the concept of filtered spaced-word matches has been utilized in multiple additional applications: The program READ-SPAM [166] computes a phylogenetic tree for a set of assembled or unassembled sequences following a similar procedure as FSWM. Prot-Spam [167] uses filtered spaced-word matches to estimate phylogenies from amino acid sequences; here, different parameter choices for the length l and the weight w of the pattern are necessary to account for the properties of proteins. The program Multi-SpaM [168] uses a combination of spaced-word matches and maximum likelihood estimation to produce phylogenies. In Multi-SpaM, blocks of consecutive spaced-word matches are searched in quadruples of input sequences and quartet trees are created from such sets. Then, all quartet trees are merged into a single super tree using the software Quartet MaxCut [169]. Slope-SpaM [170] is another novel approach to estimate phylogenetic distances based on spaced-word matches. The main idea is to estimate the sequence similarity from the slope of the function of the number of words of length k with respect to k. This function is shown to be affine linear between distinct values of k, named k_{min} and k_{max}. Using spaced words with weight k instead of contiguous k-mers improves the distance estimates. Slope-SpaM determines the values k_{min} and k_{max} for two given sequences, calculates the number of spaced-word matches with weight $w = k_{min}$ and $w = k_{max}$ and estimates their Jukes-Cantor corrected average number of nucleotide substitutions per sequence position based on the slope. Filtered spaced-word matches have also been used to detect anchor points for whole genome sequence alignments [171].

2.4.4 Applications of Alignment-free Sequence Comparison Methods

As for sequence alignments, the use cases for alignment-free methods are wide-ranging. However, their focus often lies on time and memory efficiency, making them particularly suited for data sets from recent sequencing technologies that comprise large quantities of sequences, very long sequences, or both. Accordingly, alignment-free methods have generated great interest in the field of metagenomics, where large amounts of short reads need to be processed. Here, alignment-free methods provide a useful toolbox for handling tasks such as read binning [172, 173], read assignment [174, 175], genome assembly [176], phylogenetic placement [13, 177], or visualization of metagenomic data [178]. These applications will be discussed in detail in Sec. 2.6. It is also becoming more common to apply a hybrid approach using alignment-free and alignment-based methods collectively or consecutively [179]. For example, fast alignment-free methods are initially used to screen large sequence sets for potential sequences of interest; subsequently, more accurate but slower alignment-based methods are applied to this preselection [180]. Combining alignment-free and alignment-based approaches can improve results over using only one of the approaches [181]. Alignment-free methods are also well suited in cases where sequence rearrangements are present. For this, specific programs have been developed that identify and visualize rearrangements [143, 182]. Furthermore, alignment-free methods are suited for distantly related sequences where it is challenging to construct sequence alignments [11], but some alignment-free algorithms also specifically target closely related sequences [183]. Thereby, a major use case for distances generated by alignment-free methods is the reconstruction of phylogenetic trees [161]. Moreover, alignment-free methods have also been extensively utilized in epigenomics [184], for the comparison of regulatory sequences [185], to model and predict protein families [186], or for phenotype prediction [187].

2.5 Phylogenetics

Phylogenetics centers on the analysis and reconstruction of evolutionary relationships between species, groups of species, or biological sequence entities such as single genes. The most common models to represent and visualize such relationships are *phylogenetic trees*, also called *phylogenies.*

2.5.1 Phylogenetic Trees

A graph $\mathcal{G}$ is a tuple $\mathcal{G} = (V, E)$ consisting of a set of p *vertices* $V = \{v_1, v_2, \dots, v_p\}$ (also called *nodes*) and a set of q *edges* $E = \{e_1, e_2, \dots, e_q\}$. The edges are a set of tuples $E = \big\{(v_i, v_j) \mid i, j \in [p]\big\}$. An edge (v_i, v_j) *connects* the two vertices v_i and v_j. A *path* $\rho = (v_i, \dots, v_j)$ between two vertices v_i and v_j is a list of vertices starting with v_i and ending with v_j such that there is an edge (v_k, v_l) or (v_l, v_k) in E for every two consecutive vertices v_k and v_l in ρ. If there is at least one path ρ between every two vertices of V, the graph $\mathcal{G}$ is called *connected.* If there is at most one path ρ between any two vertices of V, the graph $\mathcal{G}$ is called *acyclic.* If $\mathcal{G}$ is connected and acyclic, then $\mathcal{G}$ is called a *tree.* Only trees are considered from now on and denoted as $\mathcal{T}$. In trees, since they are connected, every vertex occurs in at least one edge of E. A node v_i that occurs in exactly one edge of E is called a *leaf node* of $\mathcal{T}$; all other nodes of V are called *internal nodes* of $\mathcal{T}$. A tree $\mathcal{T}$ is called *strictly bifurcating* if and only if every internal node $v_i \in V$ occurs in exactly three edges of E. A *clade*, or *subtree*, $V_c \subset V$ is a subset of vertices of V such that there exists a path between every two vertices of V_c by only using edges with vertices from V_c. The trees introduced so far are called *unrooted trees* — they do not specify a hierarchy of the nodes. On the contrary, *rooted* trees have an additional node v_0, called the *root* of $\mathcal{T}$, which extends the set of nodes to $V = \{v_0\} \cup \{v_1, \dots, v_p\}$. For rooted trees, the *level* of a node v_i, $i \in [p]$ is denoted as $l(v_i)$ and defined as the number of vertices in the distinct path from v_0 to v_i minus 1; the level of the root is $l(v_0) = 0$. Hence, a root v_0 of a tree $\mathcal{T}$ imposes an order on all other nodes of V where higher levels are assigned to nodes that are further away from the root. In addition to the above definition of strictly bifurcating trees, the root v_0 in a strictly bifurcating rooted tree appears in exactly two edges. For rooted trees, the *lowest common ancestor* of two leaf nodes v_i and v_j of $\mathcal{T}$ is the internal node v_k on the distinct path $p = (v_i, \dots, v_k, \dots, v_j)$ that has the lowest level $l(v_k)$. A phylogenetic tree can have a *length* associated with each edge $e \in E$, also called the *weight* of e; the weight of an edge e is denoted as $w(e)$. Thus, an edge-weighted tree is defined by its set of weighted edges $E' = \big\{(v_i, v_j, w) \mid i, j \in [p],\ w \in \mathbb{R}\big\}$. In phylogenetic trees with edge lengths, the length of a path ρ is the sum of the lengths of all edges on ρ. Edges are also referred to as *branches.* A phylogenetic tree without specified edge lengths is also called a *cladogram.* Cladograms only indicate the structure of the tree, also called its *topology.*

Typically, phylogenetic trees are defined to be strictly bifurcating and are either unrooted or rooted. A phylogenetic tree $\mathcal{T}$ represents the evolutionary relationships between a set of m species where m is the number of leaf nodes in $\mathcal{T}$. Leaf nodes are commonly labeled with m distinct identifiers for the given species. Using strictly bifurcating trees to represent phylogenetic relationships assumes that a speciation event results in exactly two new species. However, if $\mathcal{T}$ is not required to be strictly bifurcating, nodes with more than three edges are allowed; such an instance is called a *polytomy* or *multifurcation.* The existence of polytomies is either a result of an insufficient phylogenetic signal to further resolve the topology or of a divergence event of multiple species. Every internal node in a phylogenetic tree represents the common ancestor of all species in the leaves below; thus, the root is the last common

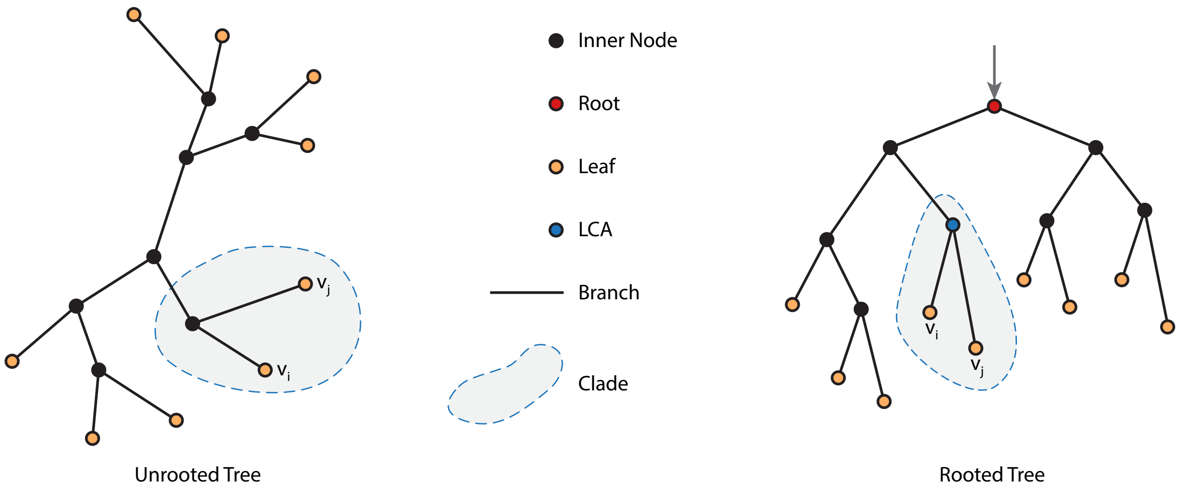


Figure 2.8 – A strictly bifurcating unrooted tree (left) and a strictly bifurcating rooted tree (right) representing the phylogenetic relationships between a set of nine unlabeled species. The edges (black lines) represent the evolutionary process. The root (red dot) in the right tree is the only node with two adjacent edges and represents the last common ancestor of all nine species. Each leaf (orange dots) represents a single species and inner nodes (black dots) represent unknown ancestor species. An example of a clade is illustrated (blue dotted line) that comprises two species v_i and v_j and a common ancestor, which is the last common ancestor (blue dot) in the rooted tree. If the root is removed from the right tree, both trees have an identical topology. The length of each edge represents its weight and indicates the amount of evolutionary change that occurred between the two connected species; the two depicted trees have different edge weights.

ancestor of all species in the tree. The edge weight $w(e)$ of an edge e encodes the estimated amount of evolutionary change that happened between the species attached to the edge. This amount of change is also referred to as *phylogenetic distance*. Similarly, for a path ρ between two leaf nodes v_i and v_j, the smaller the length of ρ the closer the species are evolutionarily related. In the optimal case, the length of a path p between any two leaves v_i and v_j labeled with the two species S_1 and S_2 is identical to the phylogenetic distance between S_1 and S_2. Figure 2.8 shows an unrooted tree and a rooted tree and highlights exemplary examples of the introduced concepts of inner nodes, leaves, a clade, and a lowest common ancestor.

The main goal within the field of phylogenetics is the *reconstruction* of a phylogenetic tree $\mathcal{T}$ given data for a set of m species. Early approaches for phylogenetic tree reconstruction were based on the presence or absence of distinct phenotypes of the species under investigation. In such a case, a $m \times R$ matrix indicates the presence or absence of R phenotypes for all species. Species that share a phenotype are assumed to be more closely related than those species that exhibit different phenotypes and therefore should appear closer together in the resulting phylogenetic tree, see Fig. 2.9 for an example. Many algorithms exist to resolve a phylogenetic tree based on a phenotype matrix, for example, by utilizing the Shannon entropy. However, the use of phenotypic features has drawbacks: It is time intensive to assess the phenotypes of a large number of species, and the presence or absence of a phenotype cannot always be identified without ambiguity. Furthermore, since the number of observable phenotypes is

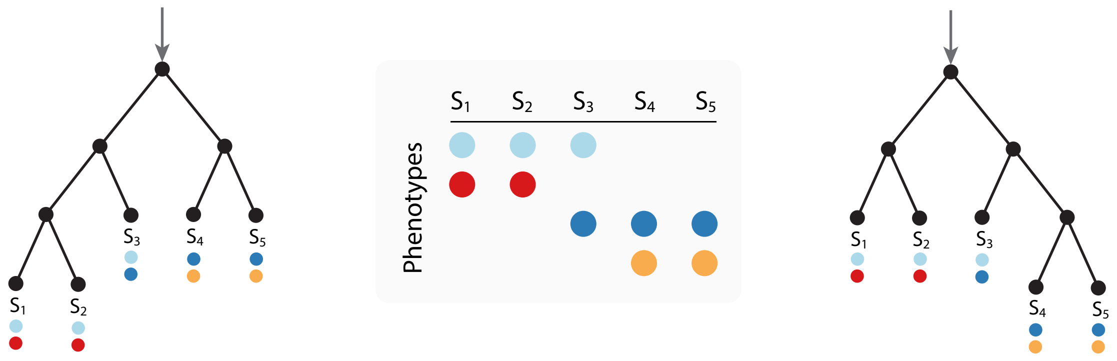


Figure 2.9 – Two possible phylogenetic trees (left and right) based on a matrix (middle) involving four phenotypes (colored circles) and five species (S_1 to S_5). The topologies of both trees represent the evolutionary relationships of the species equally well, given the limited information of the four phenotypes: Those species that share the same phenotypes are grouped together, while those species that display different phenotypes are separated. Species S_3 shares one phenotype with S_1 and S_2 and another phenotype with S_4 and S_5. Additional data is required to further resolve the correct bifurcating topology.

limited, trees cannot always be completely resolved, especially when many species are involved. Consequently, phylogenetic trees are usually reconstructed by using the genetic content of the involved species nowadays. Using genomes or parts thereof to infer evolutionary relationships is also referred to as *phylogenomics*. However, it is not obvious which parts of the genetic content can or should be used when reconstructing phylogenetic relationships and there are many pitfalls when constructing phylogenies based on genomic data [188]. For eukaryotes, there is not only chromosomal DNA but also mitochondrial DNA, and for plants additionally chloroplast DNA. All are influenced by evolution in various ways and, as such, have different advantages and drawbacks for the reconstruction of phylogenies [189–191]. But even within the chromosomal DNA, the content of the genome varies considerably between different regions, evolves at differing rates, and is subject to diverse internal and external pressures [192]. Thus, the use of different genomic regions is likely to result in different phylogenetic trees. And, more importantly, contradicting phylogenetic signals in DNA are common and have to be resolved; especially when performing phylogeny reconstruction based on the whole genome. One possibility to circumvent this problem is to deliberately choose a genomic region that suits itself to phylogenetic studies. Such a region should be conserved across all species under investigation to acknowledge its homologous nature while simultaneously providing sufficient phylogenetic signal to build a robust tree. Genomic regions typically considered for this task are single genes that are present in all involved organisms, in this context also referred to as *marker genes*. However, the choice of a marker gene largely depends on the organisms under consideration and a variety of suggestions for appropriate markers exist [193–195]. Any phylogenetic analysis that depends on a single gene inevitably reconstructs the phylogenetic history of the chosen gene, not of the species that carry it. Thus, a common distinction is drawn between *species trees* and *gene trees*. Whereas gene trees are constructed from single genes, species trees represent the combined phylogenetic information from a multitude of genes or the whole genome in a single tree. For this, any conflicting signals must be resolved; however, this is especially difficult when contradictory phylogenetic signals are supported equally well by the data. One solution is to represent complicated evolutionary relations by *phylogenetic* *networks* instead of bifurcating trees. A network $\mathcal{G}$ may contain cycles or multifurcations; every node might be present in an arbitrary number of edges, allowing organisms to have a network of adjacent organisms. As a consequence, networks represent all kinds of DNA movements between organisms in addition to inheritance, such as horizontal gene transfers or hybridization events. Biological mechanisms that cause non-tree-like relationships are also called *reticulate evolution* and can only be described by phylogenetic networks. The application of phylogenetic networks in phylogenomics has been intensively studied [196, 197] and different methods exist to infer phylogenetic networks based on maximum likelihood [198] and maximum parsimony [199]. By now, multiple tools have been developed to reconstruct and work with phylogenetic networks [200, 201] and attempts have been made to combine the results from phylogenetic trees and networks [202]. For distantly related species, it is also feasible to infer species phylogenies solely from their gene content and the order of genes [203].

Despite the fact that phylogenetic networks provide a proficient answer for reticulate evolution, phylogenetic trees often remain the primary method of choice for multiple reasons: Phylogenetic tree reconstruction methods have a long history of advancement and existing methods are well designed, easy to use, and resulting trees and auxiliary information are straightforward to interpret. In contrast, methods that reconstruct phylogenetic networks are not as sophisticated yet [192]. Due to their comparatively little use, progress remains slow and the number of available methods is limited. Phylogenetic networks are also more difficult to comprehend and more challenging to visualize, contributing to their low popularity. Also, it is not necessary to use phylogenetic networks when little or no reticulate evolution has happened in the first place. This is often the case when organisms have limited possibilities for DNA exchange other than reproduction or if they are closely related to one another. Only the construction and use of phylogenetic trees is discussed in the following. Methods for inferring phylogenetic trees from genetic content are grouped into three categories: Approaches based on a matrix of distances calculated between every pair of sequences; methods that use the *maximum parsimony* optimality criterion to infer trees; and lastly, probabilistic methods such as *Maximum Likelihood* or *Bayesian* inference.

2.5.2 Distance-Based Methods

Distance-based methods estimate a phylogenetic tree from a matrix of pairwise distances between all species that has been calculated in a preceding step. More specifically, given m sequences labeled from 1 to m, each entry $\mathbf{M}_{ij}$, $i, j \in [m]$ in a distance matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$ stores the distance between the sequences S_i and S_j. How the distances in $\mathbf{M}$ were calculated does not influence the tree reconstruction process; this makes distance-based methods notably versatile. For example, many alignment-free methods generate such distance matrices, see Subsec. 2.4, but distance matrices are also derived from alignments with distance measures such as the Hamming distance. Hierarchical clustering algorithms, in particular *UPGMA*, *neighbor joining*, and variants thereof, are among the most widely used methods for reconstructing trees from distance matrices. Here, each sequence is treated as a single cluster at the beginning and distances between clusters are set to those specified in $\mathbf{M}$. Then, to construct a phylogenetic tree $\mathcal{T}$, clusters are iteratively merged into larger ones by repeatedly grouping the two most similar clusters together until only a single one remains. The order in which the sequence clusters are grouped defines the topology of $\mathcal{T}$. Furthermore, branch lengths are inferred during each clustering step such that the resulting tree adequately represents the distances in $\mathbf{M}$. Distance-based methods differ with respect to the estimation of cluster-to-cluster distances and with respect to the calculation of branch lengths.

UPGMA The *unweighted pair group method with arithmetic mean* (UPGMA) [204] is a hierarchical bottom-up clustering algorithm to reconstruct a phylogenetic tree $\mathcal{T}$. Given m sequences, a sequence cluster $S \subset [m]$ is a subset of sequence identifiers. The set of all clusters is called $\mathcal{C}$. The distance between two sequence clusters $S, T \in \mathcal{C}$ is denoted as $d(S,T)$. Initially, every cluster comprises exactly one sequence and for two clusters $S = \{i\}$ and $T = \{j\}$ that represent the sequences $S_i, S_j\, i, j \in [m]$, respectively, the distance is set to $d(S,T) = \mathbf{M}_{ij}$. The distance between two clusters S and T is calculated as the average distance between all pairs of sequences in the clusters:

$$d(S,T) = \frac{1}{|S| \cdot |T|} \sum_{s \in S} \sum_{t \in T} \mathbf{M}_{st}\,. \tag{2.11}$$

In every step, the two sequence clusters S and T that satisfy

$$\operatorname*{arg\,min}_{S,T \in \mathcal{C}, S \neq T} \; d(S,T) \tag{2.12}$$

are merged to form a new cluster $U = S \cup T$. The distance of U to every other cluster $V \in \mathcal{C} \setminus \{S,T\}$ is updated as the average of $d(S,V)$ and $d(T,V)$ weighted by the sizes of S and T:

$$d(U,V) = \frac{|S| \cdot d(S,V) + |T| \cdot d(T,V)}{|S| + |T|} \qquad , \quad \forall V \in \mathcal{C}\,. \tag{2.13}$$

Every generated cluster corresponds to a node in $\mathcal{T}$. The primary clusters that contain single sequences are the leaves of $\mathcal{T}$ and every merged cluster U corresponds to the LCA of all sequences in U. For a newly merged cluster $U = S \cup T$, the branch lengths $w(U,S)$ and $w(U,T)$ between U and its two descendants S and T are always set to be half the distance between S and T:

$$w(U,S) = w(U,T) = \frac{d(S,T)}{2}\,. \tag{2.14}$$

Including the sizes of the merged clusters S and T for the estimation of $d(U,V)$ implies that all sequences in a cluster have equal influence on the overall cluster-to-cluster distances. Thus, the method is referred to as *unweighted* (U) PGMA. In contrast, if the newly calculated distances $d(U,V)$ are simply averaged without compensating for the sizes of the merged clusters, the algorithm is referred to as the *weighted* pair group method with arithmetic mean (WPGMA) and the distance update is defined as

$$d(U,V) = \frac{d(S,V) + d(T,V)}{2} \qquad , \quad \forall V \in \mathcal{C}\,. \tag{2.15}$$

UPGMA assumes a molecular clock and produces a tree where all leaves have the same distance from the root, a so-called *ultrametric* tree.

Neighbor Joining Like UPGMA, the *neighbor joining* (NJ) method [205] is a hierarchical bottom-up clustering algorithm but differs with respect to how inter-cluster distances and branch lengths are calculated. *Neighbor joining* does not assume that all species evolved at a uniform rate and, thus, will not create an ultra-metric tree. In each step, given the set $\mathcal{C}$ of existing clusters, two clusters $S, T \in \mathcal{C}$ are merged. For this, NJ calculates

$$d'(S,T) = (|\mathcal{C}| - 2) \cdot d(S,T) - \sum_{V \in \mathcal{C}} d(S,V) - \sum_{V \in \mathcal{C}} d(T,V) \tag{2.16}$$

and selects

$$\underset{S,T\in\mathcal{C},S\neq T}{\arg\min}\ d'(S,T) \tag{2.17}$$

to form the new cluster $U = S \cup T$. The lengths of the branches between U and the two merged clusters S and T are estimated individually by accounting for their average distance to all other clusters:

$$w(U,S) = \frac{d(S,T)}{2} + \frac{1}{2\cdot(|\mathcal{C}|-2)}\left(\sum_{V\in\mathcal{C}} d(S,V) - \sum_{V\in\mathcal{C}} d(T,V)\right), \tag{2.18}$$

$$w(U,T) = \frac{d(S,T)}{2} + \frac{1}{2\cdot(|\mathcal{C}|-2)}\left(\sum_{V\in\mathcal{C}} d(T,V) - \sum_{V\in\mathcal{C}} d(S,V)\right). \tag{2.19}$$

The distance of the new cluster U to all other clusters is updated as

$$d(U,V) = \frac{d(S,V) + d(T,V) - d(S,T)}{2} \quad , \quad \forall V \in \mathcal{C}\,. \tag{2.20}$$

Selecting the next clusters to be merged according to Eq. 2.17 is greedy: The merging always chooses those clusters that result in the smallest immediate increase in branch lengths.

Given a set of topologically different trees with defined branch lengths, the principle of *minimum evolution* (ME) always chooses the tree with the smallest total sum of all branch lengths, also called the tree size [206]. NJ is closely related to the ME approach and can be described as a variant thereof. A common group of methods to estimate branch lengths for ME are least-squares methods, but any other estimation procedure is feasible as well. One variation of ME is *balanced minimal evolution* (BME): The branch lengths of different topologies are chosen as a weighted version of the predefined distance matrix between all involved species [207]. BME overcomes the necessity to calculate branch lengths explicitly and instead directly allows the computation of the overall tree sizes. BME has been demonstrated to be statistically consistent and a software to calculate BME is available [208]. Still, listing all possible tree topologies is not feasible, and reducing the search space of tree topologies is difficult in practice. Thus, several modified implementations of NJ exist that implement advanced heuristics to traverse the space of tree topologies or to speed up computations. FastME 2.0 [209] performs NJ while also implementing nearest-neighbor interchanges to increase the probability of finding a suitable tree. BIONJ [210] improves the distance matrix by incorporating covariances estimated from alignments. Such approaches have also been extended to efficiently deal with incomplete distance matrices [211]. Furthermore, NJ was adapted to be used for the construction of species trees from a set of gene trees. For this, the distance matrix is constructed such that each entry specifies the average gene-tree inter node distances [212]. ASTRID [213] is an extension of this approach designed for fast computation when a large number of taxa are involved.

Least Squares The method of *least squares* is a popular method for formulating optimization problems. The idea is that those model parameters that minimize the sum of squared residuals are the best, whereby a residual is the difference between an observed quantity and the value provided by the parameterized model. In the case of phylogenetic reconstruction, least-squares methods search the tree $\mathcal{T}$ that minimizes the sum of squared distances between a distance matrix $\mathbf{M}$ and the distances of the branch lengths in $\mathcal{T}$. The weight of the path between the leaves labeled with sequences $i, j \in [m]$ is denoted as $d_{ij}(\mathcal{T})$. Then, the tree that minimizes the sum of squared differences

$$\arg\min_{\mathcal{T}} \sum_{i,j=1}^{m} \left(\mathbf{M}_{ij} - d_{ij}\left(\mathcal{T}\right)\right)^2 \tag{2.21}$$

is the best one according to the least-squares approach. Solving this for a single topology is straightforward and has been used in conjunction with minimum evolution [208]. However, determining the overall tree that satisfies Eq. 2.21 also involves the exploration of all tree topologies; this is challenging as there are $(2m-4)!\,/\,\left((m-2)!\cdot 2^{m-2}\right)$ different unrooted trees with m distinct leaves [214]. For any significant number of species, it is impossible to check all topologies. Instead, heuristics to explore the tree space are essential to arrive at a solution. Furthermore, the formula shown in Eq. 2.21 assumes that all distance measures $\mathbf{M}_{ij}$ are independent. When distances are derived from sequence data, the distance estimates covary as evolutionary events at inner branches of the tree effect multiple species at once. This is countered by introducing a weighting term that is incorporated into the optimization problem:

$$\arg\min_{\mathcal{T}} \sum_{i,j\in[m]} w_{ij}\left(\mathbf{M}_{ij} - d_{ij}(\mathcal{T})\right)^2 . \tag{2.22}$$

A common choice for the initialization of the weights w_{ij} is to set them as the inverse of the covariance matrix of $\mathbf{M}$, provided the covariance can be estimated reasonably well. Eq. 2.22 is referred to as *ordinary least squares*; additional least squares formulations exist that account for different biases in the data [214–216]. Most often, a *phylogenetic generalized least squares* optimization is performed in which the covariances between species sequences are estimated based on the existing knowledge of phylogenetic relationships [217].

2.5.3 Maximum Parsimony

The *maximum parsimony* (MP) criterion assumes that the tree $\mathcal{T}$ that requires the smallest number of evolutionary events to explain the provided data is the best one. Historically, the most parsimonious tree was estimated from a matrix that specified the presence or absence of different observed phenotypes of the species. In the case of molecular data, the input matrix is a MSA of the reference sequences where the features are nucleotides (or proteins) of the aligned sequence positions. Minimizing the number of evolutionary events implies that the tree generated from MP has the least amount of homoplasy possible. This idea is comparable to Occam's razor, as it assumes that the most simple model is the preferable one. It also has certain parallels to the notion of minimum evolution; however, unlike minimum evolution, MP has been shown to be inconsistent [218]. Also, if homoplasy is present in the data, the fundamental assumption of MP is violated and the amount of divergence between species is systematically underestimated as a result; this problem is also known as *long-branch attraction*, as very divergent species can be grouped together by MP even though they belong to different evolutionary branches.

Without loss of generality, it is assumed that the tree is reconstructed on the basis of a MSA $\mathcal{A}$. Solving MP consists of two sub-problems: the *small parsimony* problem and the *maximum parsimony* problem. The small parsimony problem is defined as finding the minimal number of evolutionary changes required to explain $\mathcal{A}$, provided a fixed topology of $\mathcal{T}$. The positions of $\mathcal{A}$ are assumed to be independent from each other. Consequently, it is possible to calculate the small parsimony problem for each alignment column independently and sum the minimal number of inevitable substitutions over all columns. For each individual column, this is carried out efficiently by traversing $\mathcal{T}$ from the leaves toward the root and noting the set of symbols at every internal node that minimizes the number of mutations in the clade below via dynamic programming [219, 220]. The MP problem finds the optimal overall tree topology for $\mathcal{T}$ such that the number of evolutionary events is minimized. In contrast to the small parsimony problem, there is no known method to efficiently reach the optimal solution to the MP problem, as it is NP-complete [221]. Determining the most parsimonious tree requires to list all trees and calculate their parsimony score, which is practically impossible for large sets of species. Instead, algorithms that perform phylogeny reconstruction with MP utilize heuristics that only traverse parts of the complete tree space; for this, they start with an initial tree topology and gradually explore *neighboring* tree topologies with improved parsimony scores. The neighboring topologies are obtained via different operations on the tree topology such as *nearest-neighbor interchanges*, *subtree-pruning and regrafting*, or *subtree bisection and reconnection*. The accuracy of morphologically-based MP has been studied in detail as MP provides a straightforward framework to generate trees from phenotypes [222]. Several universal software programs support MP estimation of phylogenetic trees [223–225] and MP has also been combined with the maximum likelihood method under a simple model of site substitution [226]. However, the complexity of different approaches to the small and maximum parsimony problems varies greatly [227]. Recently, a fast MP algorithm has been released that also estimates branch support by bootstrapping [228].

2.5.4 Maximum Likelihood

In general, the likelihood function for a fixed data set D and a *hypothesis* H, also called the *model*, is defined as

$$\mathcal{L}(H) = P\big(D \mid H\big)\,. \tag{2.23}$$

$\mathcal{L}(H)$ is the probability of observing the data D under the assumption that H holds. However, the likelihood function $\mathcal{L}$ is not a probability measure when D is fixed and H varies and there are infinite potential hypotheses H. *Maximum likelihood* (ML) estimation searches for the hypothesis H that explains the data D best. Thus, it solves

$$\arg\max_{H}\ \mathcal{L}(H) \tag{2.24}$$

to obtain the hypothesis that maximizes the likelihood function. For ML-based phylogeny reconstruction, the model consists of the tree topology $\mathcal{T}$, its branch lengths b, and the specified evolutionary model $M(t)$ as introduced in Subsec. 2.3.1. The data is typically a MSA $\mathcal{A}$, thus:

$$\mathcal{L}(H) = P\big(\mathcal{A} \mid \mathcal{T}, b, M(t)\big)\,. \tag{2.25}$$

Here, $\mathcal{A}$ comprises m sequences over n sites. Furthermore, it is assumed that the aligned positions of $\mathcal{A}$ are independent of each other. Hence, as in maximum parsimony, the likelihood can be calculated over all aligned positions independently and then multiplied over all columns.

Computing the likelihood for a given tree is performed with dynamic programming, traversing the tree from the leaves toward the root. Following the notation introduced in Subsec. 2.3.1, alignments over the alphabet Σ are considered and the probability that symbol $\sigma_i \in \Sigma$ changed to symbol $\sigma_j \in \Sigma$ in time t is denoted as $M_{ij}(t)$ with rate r. When considering a single column $u \in [n]$ of the alignment and an internal node A of $\mathcal{T}$, L_A is defined to be the set of all leaves below A. Furthermore, $P_u(L_A \mid \sigma_i)$ is the probability of observing the nucleotides of column u of the sequences at the leaves below A when the symbol σ_i is present at node A. The calculations are performed from the leaves upward: Thus, for the two children nodes B and C of A, it is assumed that $P_u(L_B \mid \sigma_j)$, $\forall \sigma_j \in \Sigma$ and $P_u(L_C \mid \sigma_j)$, $\forall \sigma_j \in \Sigma$ are already known. Furthermore, the branch lengths between A and its children are denoted as b_{AB} and b_{AC}, respectively, and $P_u(L_A \mid \sigma_i)$ is calculated as

$$P_u(L_a \mid \sigma_i) = \left(\sum_{\sigma_j \in \Sigma} M_{ji}(r \cdot b_{AB}) \cdot P_u(L_B \mid \sigma_j) \right) \cdot \left(\sum_{\sigma_j \in \Sigma} M_{ji}(r \cdot b_{AC}) \cdot P_u(L_C \mid \sigma_j) \right) . \tag{2.26}$$

Intuitively, this computation sums up the probability over all configurations of possible states of symbols from Σ at the child nodes B and C. For each leaf node F annotated with symbol σ_i and its corresponding aligned sequence S_F

$$P_u(L_F \mid \sigma_i) = \begin{cases} 1 & \text{if } \sigma_i = S_F[u] \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

is defined. The overall likelihood for the alignment column $u \in [n]$ at the root R of $\mathcal{T}$ is inferred as

$$\mathcal{L}_u = \sum_{\sigma_i \in \Sigma} \pi_i \cdot P_u(L_R \mid \sigma_i) \tag{2.28}$$

whereas π_i is the base frequency of symbol $\sigma_i \in \Sigma$. The likelihood over all aligned columns in $\mathcal{A}$ is then calculated as the product of the likelihoods for all single columns:

$$\mathcal{L} = \prod_{u=1}^{n} \mathcal{L}_u .$$

While maximum parsimony considers only a single combination of symbols at inner tree nodes, ML takes all possible configurations of symbols at ancestral sequences into account. Due to computational limitations when calculating the product of small numbers, the logarithm of the likelihood is normally considered instead:

$$\mathcal{L}_{\log} = \sum_{u=1}^{n} \log \mathcal{L}_u .$$

This algorithm [229] does not take into account that evolutionary rates vary across different sites, but there are adjusted versions that do [230]. Although long branch attraction is mostly associated with maximum parsimony, it can also affect ML trees [231]. However, it is thought that the effects are not as pronounced when using maximum likelihood [232]. It is mandatory though to choose an evolutionary model $M(t)$ in order to estimate trees with ML. There are a variety of models to choose from and there are different strategies to select the best model for the data at hand [233]. However, a recent study also suggests that sophisticated model selection is not necessary, and instead it is best to always employ the model with the most parameters [234]. As for other tree reconstruction approaches, a variety of programs exist that are tailored to the calculation of ML trees [235–237]. Furthermore, ML is used to infer phylogenetic networks as well [238]. ML is currently the most popular and trusted approach to reconstruct phylogenetic trees and several large-scale studies applied ML to generate phylogenetic trees that span whole kingdoms or phyla of species, for example, for vertebrates [239], fungi [240], or bacteria and archaea [241].

2.6 Genomics and Metagenomics

As explained in Sec. 2.1, the order of nucleotides in DNA encodes hereditary information and is responsible for the complexity of all functioning organisms. In pursuit of understanding life as we know it, assessing similarities and divergences in different DNA sequences is essential and often among the first tasks in any bioinformatic analysis. But first, the order of nucleotides in a DNA sequence must be determined, a process called *sequencing*. A wide variety of sequencing methods exist and the appropriate method has to be chosen with respect to the available biological sample and the specific bioinformatic task planned. The sequencing technique strongly influences the potentials and limitations of subsequent data processing steps.

Analyzing genomes of *deliberately selected* organisms is called *genomics*. In contrast, *metagenomics* is the study of the entirety of all genomic data present in an environmental sample. Thus, it considers organisms that are directly recovered from the environment, often with the goal of characterizing the environment and not the organisms themselves [6, 242]. The concept was first termed the *metagenome* of soil [243]; but later, the term metagenomics was introduced to deal with biological sequencing data from environmental samples in general [244]. The rise of metagenomics enabled the insight that most of microbial life has been missed by earlier sequencing approaches due to methodological restrictions [6]. The sequencing technologies used for metagenomics aim to reproduce a complete portrayal of all organisms present in the environment. However, biases regarding the sequenced organisms still occur frequently [245].

Resulting sequence information takes on different forms and may be used in many contexts: One of the primary questions in metagenomic studies is to determine the presence or absence of species in the environment, as well as their respective abundances. If properly sequenced and processed, metagenomic data can also answer questions regarding the influence of environmental factors on the species community, the interaction of organisms with the environment, the interaction of organisms with each other, or the functional abilities of the species community [246]. Hence, metagenomics is also referred to as *environmental genomics*.

A metagenomic study comprises many consecutive steps, called a *pipeline*, which can be divided into two parts: The first part comprises the sampling effort in the environment, the sample preparation, and the sequencing itself. The second part includes the computational steps in which the sequenced data is cleaned and analyzed. To understand the intricacies of metagenomic approaches, it is mandatory to reflect on both of these parts. Thus, the following sections first discuss common sequencing techniques and then a selection of research areas and their computational steps within the field of metagenomics.

2.6.1 DNA Sequencing

Methods for DNA sequencing are commonly grouped into three technological eras that correspond to their time of development and coincide with their methodological approach to sequencing [247, 248]. As a consequence, they differ with respect to their sequencing speed, their sequencing accuracy, the resulting number of sequences, and the length of these sequences: *First-generation* DNA sequencing methods are characterized by the sequencing of clonal DNA with relatively low throughput. *Second-generation* sequencing added large parallelization to sequencing procedures, resulting in high sequence throughput in little time. *Third-generation* sequencing is marked by techniques that are able to sequence single molecules, often with great length.

In the early stages of sequencing, the *first-generation technologies*, only single species could be sequenced via clonal approaches [249]. In these approaches, the target sequence molecule to be cloned is combined with a sequence molecule of a living host, called the *vector DNA*, to form *recombinant DNA* (rNDA). The rDNA is inserted into the living host organism, which is then repeatedly replicated resulting in a large population of host organisms, all of which contain identical rDNA with the target DNA [250]. The first widespread sequencing method based on clonal populations is known as Sanger sequencing [251, 252] and was used to sequence the first DNA genome in 1977 [253]. Sanger sequencing follows the *sequencing-by-synthesis* approach where a DNA polymerase synthesizes a complementary DNA strand to the given DNA template. Modified chain-terminating nucleotides halt the synthesis when incorporated into the new DNA strand. Additionally, the modified nucleotides may be either radio- or fluorescently-labeled. Thus, it is possible to detect which molecule is incorporated into the sequence at each step and the complete sequence of the DNA template molecule is reconstructed. The detection step was first performed manually by electrophoresis on polyacrylamide gels before being automated, resulting in the first commercial sequencing machines [254].

Sanger sequencing, although very accurate, is a time-intensive approach where only sequencing speeds well below one megabase per hour are reached [255, 256]. The produced sequences have a length of up to 1000 bp [247]. Another first-generation technique is Maxam-Gilbert sequencing [257]. Instead of sequencing-by-synthesis, Maxam-Gilbert sequencing breaks a DNA molecule of radio-labeled nucleotides into fragments by treating it with chemicals. Again, the order of nucleotides is determined when running the resulting fragments with different lengths through a polyacrylamide gel [249]. Further scientific advances such as the polymerase chain reaction (PCR) [258] greatly accelerated the emergence of improved sequencing protocols. However, both Sanger and Maxim-Gilbert sequencing rely on cultured organisms or PCR-amplified DNA, posing a severe limitation to their applicability: While microorganisms account for the majority of living life on the planet [259], the vast majority of them cannot be cultured. Thus, by only using the Sanger or Maxam-Gilbert method, most organisms cannot be sequenced and remain unknown.

Therefore, several improved sequencing methods emerged that are generally termed second-generation or next-generation sequencing (NGS) techniques. These methods were developed in the first decade of the twentieth century, a period also referred to as the sequencing revolution [260]. NGS techniques share the property of generating a large number of short sequence fragments of the DNA or RNA present in a sample, often by performing sequencing steps in a greatly parallelized manner. These sequence fragments are commonly referred to as *reads*. *Pyrosequencing* was one of the first next-generation techniques and also follows the sequencing-by-synthesis paradigm [261, 262]. However, in comparison to Sanger sequencing, nucleotides are detected via luminescence during pyrophosphate synthesis. To detect the next nucleotide in the sequence, four solutions containing one of the nucleotides are added sequentially to the to-be-sequenced DNA. If a nucleotide is incorporated into the template strand by the DNA polymerase enzyme, light is emitted and recorded. This approach was highly parallelized in 2005 and led to the first commercially available next-generation sequencing platform [263]. Another group of next-generation sequencing techniques is known as Solexa sequencing [264] or Illumina sequencing [265] and uses a sequencing process called *bridge amplification*. Compared to other techniques, bridge amplification enables the possibility to sequence a given DNA strand from both directions. This has been termed *paired-end* sequencing and has several potential advantages for the accuracy of post-processing steps [247], see Subsec. 2.6.2. Subsequently, Illumina developed a variety of sequencing platforms, all with different properties regarding sequence accuracy, read lengths, and sequencing speed [266]. Furthermore, a commonly used technology is the *sequencing by oligonucleotide ligation and detection* (SOLiD) system [267] which does not conform to the principle of sequence-by-synthesis, but instead utilizes a sequence-by-ligation approach. The mentioned sequencing technologies led to a large improvement in sequencing speed and accuracy [268]. Current Illumina systems can sequence more than 6 gigabases per hour, resulting in unprecedented sequencing speeds [269]. Additionally, compared to first-generation sequencers that are dependent on clonal cell populations, second-generation sequencers utilize DNA isolated directly from environmental samples [270]. This paradigm change had a large impact on genomics and put forth the field of metagenomics as it is today [271]. The emerging data also entailed large computational demands for the post processing of the sequencing data, and thus algorithms had to scale with sequencers. Especially in the area of biodiversity studies, many new algorithmic techniques were developed that focus on massive amounts of data with increasing complexity [242].

Unfortunately, sequencing errors are an integral part of all sequencing platforms and, hence, every metagenomic study. Sequencing errors are introduced not only during sequencing itself, but also during handling of samples in the wet lab, for example, during library preparation and PCR enrichment [272]. Second-generation sequencing platforms usually provide an estimate of their accuracy, the *error profile*. Error profiles are important for quality control and filtering of generated reads [273, 274]. All commonly used first- and next-generation techniques use short DNA fragments, called *primers*, to initiate the sequencing reaction. Thus, the choice of primers determines at which position the DNA sequencing starts in the first place, see Subsec. 2.6.2.

In recent years, another type of sequencing methods has been developed; they are characterized by the property to produce *long* reads at fast paces. Another feature of these methods is their ability to sequence single DNA molecules. These methods are commonly referred to as third-generation sequencing techniques, although there is some debate on their defining features [247]. Pacific Biosciences developed the first techniques commonly attributed to third-generation sequencing, namely the single-molecule real-time platform [275]. In contrast to second-generation sequencers, the reads produced by these sequencers are orders of magnitude longer. The biggest competitor today in the area of third-generation sequencing is the *Nanopore* sequencing platform. In contrast to all previously developed techniques that depend on either PCR amplification or chemical labeling, nanopore techniques require neither. Instead, base detection of DNA is performed by measuring changes in electrical current in an electrophoresis solution while the DNA strand passes through a nanopore in a membrane [276]. While the method cannot yet produce reads as accurate as other methods, it has the ability to create reads with lengths above one megabase; more common are read lengths between 10 and 30 kilobases though [277]. One of the available sequencers is also the smallest sequencing machine created so far with a size similar to that of a USB flash drive that allows sequencing directly in the field [278, 279]. By now, there is a wide range of long-read sequencers [280] and comprehensive reviews about their trade-offs exist [281]. Long reads with high error rates are a great gain for the area of genomics and metagenomics, but the opportunities also come with great challenges [277]. On the one hand, there are numerous applications that benefit greatly from long-read techniques, especially for de-novo assembly of unknown genomes or when dealing with human genomes [151, 282]. On the other hand, NGS techniques are still equally important for a wide range of applications where high accuracy is necessary. Ultimately, the proper choice of sequencing techniques depends on the research question at hand.

2.6.2 Metagenomic Methods

There has been some debate about the accurate designation of different methodological approaches in metagenomics. Here, the vocabulary proposed by Marchesi and Ravel [283] and extended by Breitwieser et al. [7] is used to differentiate between three essential approaches: *Metataxonomics*, *metagenomics*, and *meta-transcriptomics*, which are discriminated primarily with respect to the type of sequence data under study. Metataxonomics is only based on sequencing data from *specific* genetic regions of organisms, called *marker genes* or *marker regions.* Marker genes are specifically selected for the task of identifying organisms present in an environmental sample. Typical marker genes exhibit sufficient variability to differentiate between a large number of species. However, their flanking regions are highly conserved, which allows the design of specific primers in order to retrieve only the desired genes from an environmental sample. Depending on the organisms of interest, different genes are frequently used as markers: Among the most common are the 16S rRNA for prokaryotes, the 18S rRNA for eukaryotes, and the ITS region for fungi [193, 194]. ITS has also been used for plants, especially algae [195]. Since these genes act like a barcode for species, metataxonomics is also commonly referred to as *metabarcoding* or *amplicon sequencing.* In contrast, following the conventions of Breitwieser et al., the term metagenomics specifically refers to shotgun *whole genome sequencing* (WGS). Here, the phrase 'shotgun' is used to refer to the fact that the sequencing effort is not targeted to specific regions such as marker genes but yields DNA reads across all available genomes at random. For eukaryotes, this includes not only the chromosomal DNA but also the mitochondrial DNA and for plants also the chloroplast DNA. Thus, amplicon sequencing in metataxonomics and shotgun WGS sequencing in metagenomics cater to fundamentally different needs and must be employed accordingly. Meta-transcriptomics studies messenger RNA, the transcribed DNA, extracted from the environment. The resulting sequences provide information on genes that are actively transcribed in a species community. It also provides an opportunity to investigate the expression levels of the active genes. Other congruent approaches to examine community-based molecular processes include metaproteomics and meta-metabolomics [7]. Here, the focus is primarily on metataxonomics and metagenomics, thus, sequencing data obtained either from targeted amplicon sequencing or from shotgun WGS sequencing.

According to the definitions introduced, the sequencing data present in metagenomic studies are short sequencing reads, commonly produced by next-generation sequencing techniques. The short reads obtained from the sequencing machine contain errors and biases from the sampling steps, the wet lab preparation, or the sequencing itself. Thus, the first step in metagenomic pipelines is the trimming of reads and filtering out erroneous reads, reads that are too short, or reads that exhibit other unusual behaviour. Hereby, error profiles provided by the sequencing platforms guide the process of quality control. Many programs are available that perform read trimming [284, 285] and quality control [286, 287], resulting also in a wide variety of pre-packaged pipelines for these tasks [288, 289]. The subsequent objectives strongly diverge depending on the goal of the analysis; several reviews offer an overview of the potential steps performed in metagenomics [6, 246] and the most prevalent of them are introduced here.

Assembly is the process of constructing longer sequence segments from short reads, a common task in metagenomics. The objective is to concatenate short reads that originate from the same genome into fragments that are as long as possible. The assembled fragments are called *contigs.* A collection of contigs that covers a significant proportion of a genome is referred to as a *scaffold.* In the optimal case, the assembly of short reads results in the reconstruction of whole genomes of one or more species present in the sample. This process is called *de-novo assembly* and the resulting genomes are called *metagenome-assembled genomes*

(MAGs). The quality of assembled genomes is highly dependent on the sequence *coverage*; here, the coverage of a genome in a given set of short reads is a measure of the prevalence of the genome within the reads. More specifically, it is defined as the number of times a position of the original genomic sequence is present in a read averaged over all sequence positions. The higher the coverage of a genome, the easier it is to assemble long contigs, scaffolds, or even the complete genome. The de-novo assembly of a previously unknown sequence from a set of short sequencing reads is NP-hard [290] and, thus, no feasible solution exists to construct the correct solution in an acceptable time frame. However, heuristic methods perform the task reasonably well. The most widely implemented strategies are greedy methods, overlap-layout consensus approaches, and De Bruijn graph assemblers based on k-mers extracted from the reads [291]. Popular programs for sequence assembly that follow these principles are (META)SPADES [292, 293] and SKESA [294]. Since the rise of third-generation sequencing techniques, it is becoming evident that combining short reads from NGS techniques with long reads from new sequencers greatly improves the resulting assemblies and novel programs are released that pursue such a hybrid approach [295, 296]. A more detailed summary of assembly methods can be found in the literature [291, 297].

Reads obtained in metagenomic and metataxonomic studies are inherently of unknown taxonomic origin. The identification of the taxonomic or phylogenetic affiliation of reads is a key step; however, the means by which this identification is performed vary. In general, this process is termed *taxonomic profiling* of reads or *read assignment*, see Subsec. 2.6.3. Taxonomic profiling may not only be concerned with the question *which* species are present in a given environmental sample, but also *how abundant* the present organisms are. The most straightforward approach to taxonomic profiling is to label each read individually based on a comparison with reference databases. However, it is often preferred to first *group* the reads. This reduces computational overhead in assigning taxonomic labels to the vast number of reads that are generated in metagenomic experiments. The taxonomic grouping of reads is also referred to as *binning* and the resulting groups as *bins* [298]. Subsequently, each bin is assigned a joint taxonomic label. Another approach to taxonomic profiling that yields more specific information about individual reads is *read mapping*: The reads from metagenomic experiments are aligned against reference databases to derive their exact location within a reference genome [299]. With such information, it is possible to transfer existing annotations from the references to the metagenomic sequences. This may include information about which genes are present in a metagenomic data set (*gene annotation* or *gene calling*) or about the function of present genes (*functional annotation*). Alternatively, gene annotation is performed for assembled contigs [300] or does not rely on alignments [301]. Taxonomic profiling can also be performed based on contigs [302]; while this requires a computationally expensive assembly and hence high read coverage, taxonomic labels derived from contigs are usually more accurate due to their increased length.

In metataxonomics, reads are usually grouped into operational taxonomic units (*OTUs*) or amplicon sequence variants (*ASVs*). OTUs group sequence reads that share a significant sequence similarity that is often set at 97%. This grouping drastically reduces the number of reads that have to be labeled. However, some information about closely related species might get lost in the process, especially when only a single taxonomic label is specified for each OTU. Common programs for OTU analysis include MOTHUR [303, 304] and QIIME [305]. When ASVs are created, supposedly erroneous reads are removed and the remaining ASVs represent sequences with single nucleotide differences. Common methods to create ASVs from metagenomic experiments include DADA2 [306], UNOISE [307], or DEBLUR [308]. All of these approaches come with advantages and drawbacks and comparative studies have been conducted for varying methods [309]. The comparison of metagenomic samples among each other is called *comparative metagenomics*. Often, the compared samples stem from different sample locations, or originate from the same location but were taken at different points in time. The comparison of metagenomes may involve the qualitative or quantitative species content [310], the functional characteristics of the metagenomes [311], or their environmental differences. Due to the amount of sequencing data present in metagenomics, the approaches mentioned above often follow alignment-free procedures; however, alignment-based techniques are used if high accuracy is mandatory. In particular, the problem of read assignment is among the most common problems in metagenomics.

2.6.3 Read Assignment

One fundamental task in metagenomics is the grouping of short reads based on their similarity, called *binning*. Binning can be performed reference-free or reference-dependent [312]: In the former case, single reads are mapped against genomes in a reference database. Subsequently, they are grouped according to the references they were mapped to. In the latter case, the short reads are directly clustered into groups without the help of external references. The task of *taxonomic read assignment*, also known as *metagenomic classification*, is closely related to binning and is an equally integral part of many metagenomic analyses. In taxonomic read assignment, a distinct taxonomic label is assigned to each read of a sequencing sample. Solving one of the two tasks also entails a possible solution for the other one: When reference-based binning is performed, the resulting bins simultaneously yield a taxonomic classification of the reads by means of the labeled reference sequences. Vice versa, every form of taxonomic classification also groups reads with respect to their assigned taxonomic labels and, thus, results in a collection of bins.

All methods for read assignment depend on an existing database of known organisms that are already taxonomically labeled and serve as a *reference*. For every read, the reference database is searched for similar sequences and their taxonomic labels are transferred to the input reads. In this context, reads are also called *query sequences* as they are queried against the database. As a result, every approach to taxonomic labeling is inherently dependent on the size and quality of the underlying reference database [313]. Generally, the more sequences are available in the reference database and the more accurate their taxonomic labels are, the better is the taxonomic identification of the queries [313]. However, if the reference database grows in size, there is also a necessity to store and query it efficiently to be able to process the large amounts of reads produced in metagenomic studies. Furthermore, most organisms in environmental samples may not be available in reference databases, as discussed above [288]. Additionally, the number of sequenced organisms is growing rapidly and reference databases have to be updated regularly, both on their official channels as well as on the machines of end users. All of the issues above pose serious limitations on algorithms for metagenomic classification which have to be adequately addressed when performing taxonomic read assignment. Thus, taxonomic classification methods differ with respect to their underlying database, the sequence similarity search method used to query the reads against the database, and the manner in which a distinct taxonomic label is selected from the most similar database entries. Several reviews on methods for read assignment are available [7, 288] and Fig. 2.10 shows the typical process of read assignment.

One straightforward approach to performing read assignment is the use of alignments to find similar sequences in a reference database. Such a procedure can be easily implemented with BLAST on a comprehensive database such as RefSeq [314]. The probability to discover

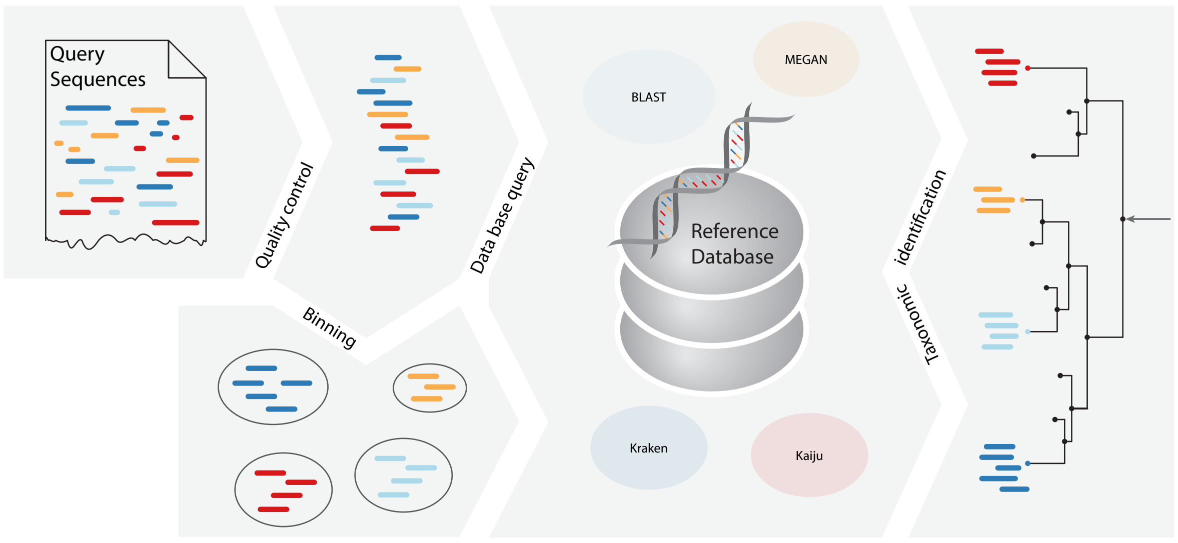


Figure 2.10 – A typical pipeline for the task of read assignment: The input is a set of query sequences from a metagenomics experiment. The reads are then preprocessed, potentially involving steps such as demultiplexing, trimming, and quality control. Subsequently, the cleaned reads are either grouped into bins or directly queried against a reference database. Taxonomic labels for each read or bin are inferred by algorithms such as BLAST, MEGAN, KRAKEN, or KAIJU. The output is a taxonomic labeling and grouping of the query reads.

a closely related sequence by a BLAST-based taxonomic assignment is high, however, the speed of the assignment does not scale with the amount of sequencing data produced in second-generation sequencing techniques. On the one hand, this is due to the algorithmic complexity of the necessary alignment step, on the other hand, due to the large size of the reference databases BLAST is usually associated with. Another more fundamental problem in the use of large-scale databases lies in the fact that they undergo little editorial monitoring. As a consequence available reference sequences are, potentially, annotated insufficiently or even wrongly and inferred taxonomic labels are unreliable. In addition, querying a single read often causes hundreds or thousands of equally likely hits with varying annotations and it remains unclear how a single taxonomic label can or should be selected in such cases. MEGAN, the METAGENOME ANALYZER [315], was one of the first general algorithmic frameworks developed to overcome the mentioned limitations. It builds upon a reference-based taxonomic assignment algorithm, such as BLAST, and adds additional functionalities: MEGAN proposes a simple algorithm to select a singular taxonomic label from all database hits: each query read is assigned to the lowest common ancestor of all sequences that emerged from the database search. Furthermore, MEGAN provides comprehensive tools for the evaluation and visualization of generated results. MEGAN is still actively maintained and additional features have been released recently [316, 317]. Since then, a multitude of programs with varying trade-offs have been designed to perform read assignment: PHYMM and PHYMMBL [318] utilize Markov models for read assignment and are specialized for short reads down to 100 bp. DIAMOND [319] uses a double-indexing structure for protein alignments to greatly improve the speed over BLAST while maintaining its sensitivity. Certainly, the recent rise of commercially available sequencing platforms that produce long reads has also caused a demand for adapted algorithms. METAMAPS [320] is specifically designed for such long-read data and performs accurate read assignment as well as estimation of sample composition.

It does not calculate exact alignments, but instead it utilizes a minimizer strategy [321] to find, for a given read, the most promising locations where it might occur in the references; from this, alignment locations and sequence identity are estimated. MEGAN has also been adapted to handle longer reads [317].

In contrast to all these methods, an entirely different approach to read assignment is to determine taxonomic labels of reads from their composition of k-mers. The idea is that the presence or number of k-mers in a read carries sufficient information to infer its taxonomic affiliation by comparing it to a pre-computed database of k-mer compositions from reference sequences. One of the first approaches to pursue this paradigm was KRAKEN [148]. KRAKEN creates a database that maps k-mers to specific taxonomic annotations as follows: Based on a set of reference sequences, KRAKEN extracts all words of length k from the sequences. For each k-mer, it finds the taxonomic label of the last common ancestor of all references that share this k-mer and associates it with the corresponding k-mer; these k-mer-label-pairs constitute the reference database. For a given query read, all k-mers of the query are matched against the reference database; each k-mer of the query casts a vote for the associated node within the taxonomy of all references, resulting in a tree structure of taxonomic labels in which nodes are weighted by the number of k-mers that voted for the respective node. The taxonomic tree is then traversed from its root towards the leaves, following the highest weighted path of votes until the lowest node that still has a weight larger than zero. The taxonomic designation of this node is used to label the query sequence. After building the reference database of k-mers once, the computational effort to infer a taxonomic label for a read is low. As a result, KRAKEN achieves high speeds while maintaining decent accuracy for the resulting taxonomic labels. Since its release, KRAKEN has received several updates with further improvements: KRAKEN 2 improves the classification speed and memory requirements by enhancing data storage techniques [322]. KRAKENUNIQUE utilizes the insight that the use of unique k-mers reduces false-positive identifications in a metagenomic setting [323]. KAIJU performs taxonomic classification by first translating DNA or RNA reads to amino acid sequences before it searches a reference database for maximum exact matches, a technique which, in contrast to k-mers, also allows mismatches [175]. A more versatile approach is taken by MMSEQS2 (Many-against-Many sequence searching) [324]: MMSEQS2 is used similarly to BLAST to query sequences against large biological sequence databases; however, it is orders of magnitude faster by combining three search stages of increasing sensitivity. The first stage utilizes a fast word-based search that uses long non-exact words. Subsequently, these non-exact matches serve as seeding areas in the next two stages in which increasingly accurate alignments with references are created. Other tools combine multiple steps from typical bioinformatics pipelines or are designed for specific types of data. For example, BRACKEN [325] is an extension of KRAKEN to estimate abundance profiles of microbial communities from metagenomic data, and METAKALLISTO [326] combines read assignment of metagenomic studies with abundance estimation of transcripts in RNA-Seq data. Several benchmark studies for taxonomic classification tools exist [327, 328]. Depending on the type and amount of accessible data, read assignment comes at a price: First, its accuracy is largely dependent on the availability and quality of reference sequences that are present in reference databases. Only when closely related reference sequences are available for all query sequences, a high accuracy can be achieved. If there are no closely related references for a query it will be either not classified or misclassified. Second, a classification based on taxonomic labels is potentially misleading when interpreted in an evolutionary context. One possible approach to avoid these difficulties is by means of *phylogenetic placement*.

2.7 Phylogenetic Placement

In *phylogenetic placement* (PP), in addition to the reference sequences themselves, their evolutionary history is provided by means of a phylogenetic tree, called the *reference tree.* A query sequence is placed within the reference tree, and thus one can derive its phylogenetic relationship to the references directly. Phylogenetic placement solves the problem of unclassified or misclassified query sequences: Queries that are closely related to reference sequences are placed close to the leaves of the tree. In contrast, queries without closely related references are placed towards the root instead. By this, it is immediately apparent for which queries no closely related references exist while their evolutionary context remains evident.

The input data for PP consists of a set of m reference sequences, a reference phylogeny $\mathcal{T}_{\text{ref}}$ with m leaves labeled with the reference sequences, and a set of query sequences. If an alignment of the references is available, it is denoted as $\mathcal{A}_{\text{ref}}$. The query sequences are to be placed within the reference phylogeny by specifying a precise location: For every query sequence S_q an existing branch e_q of $\mathcal{T}_{\text{ref}}$ is selected and divided into two branches e_{q1} and e_{q2} by a new node n_q. The two new branches e_{q1} and e_{q2} are referred to as the *proximal* and *distal* branches, respectively, whereas the distal branch is located towards the leaves of the tree. The node n_q is connected to a new leaf n'_q with a new branch e'_q and the leaf is labeled with the query sequence S_q, see Fig 2.11. The branch e'_q is referred to as the *pendant* branch. Thus, a placement position of a query S_q is uniquely defined by the insertion edge e_q, the distal branch length, and the pendant branch length. The proximal branch length is defined by and inferred from the distal branch length. Based on the specified placement position for each query S_q, a new phylogenetic tree with $n+1$ organisms — n references and the query S_q — *could* be reconstructed, resulting in a *placement tree* denoted as $\mathcal{T}_q$. However, the reference tree is generally not altered during PP. Instead, the placement position of each query is stored by specifying the placement branch e_q, the length of e'_q, and the length of the distal branch. All placement positions are typically recorded in a single file following the *JPlace* format [329].

2.7.1 Approaches to Phylogenetic Placement

There are two main categories of algorithms for phylogenetic placement: character-based methods and distance-based methods. The former are based on alignments between query and reference sequences and infer ML values in order to find placement positions. The latter methods calculate distance measures between query and reference sequences, which are then used to infer adequate positions in the reference tree. ML-based approaches to PP require multiple alignments: In general, for every query sequence S_q that has to be placed, a multiple sequence alignment comprising S_q and all m reference sequences is mandatory. In practice, the procedure is as follows: First, the MSA $\mathcal{A}_{\text{ref}}$ of all m references is built. Second, a query sequence S_q is aligned *against* $\mathcal{A}_{\text{ref}}$ only for the purpose of inferring its placement position. For this, S_q is temporarily added to $\mathcal{A}_{\text{ref}}$ and removed again after its placement so that S_q does not influence other query sequences. This procedure assumes that the query at hand fits well into the structure of the predefined reference alignment. If the query possesses long insertions or many mutations, its alignment against the references may be sub-optimal and the accuracy of the phylogenetic placement is effected negatively.

Two of the first approaches that performed phylogenetic placement were EPA [10] and PPLACER [12]. Both algorithms are based on the maximum likelihood principle to determine optimal placement positions, and their underlying algorithm is very similar: Each query sequence is placed at every internal branch of $\mathcal{T}_{\text{ref}}$. Thus, for a tree with m references, there

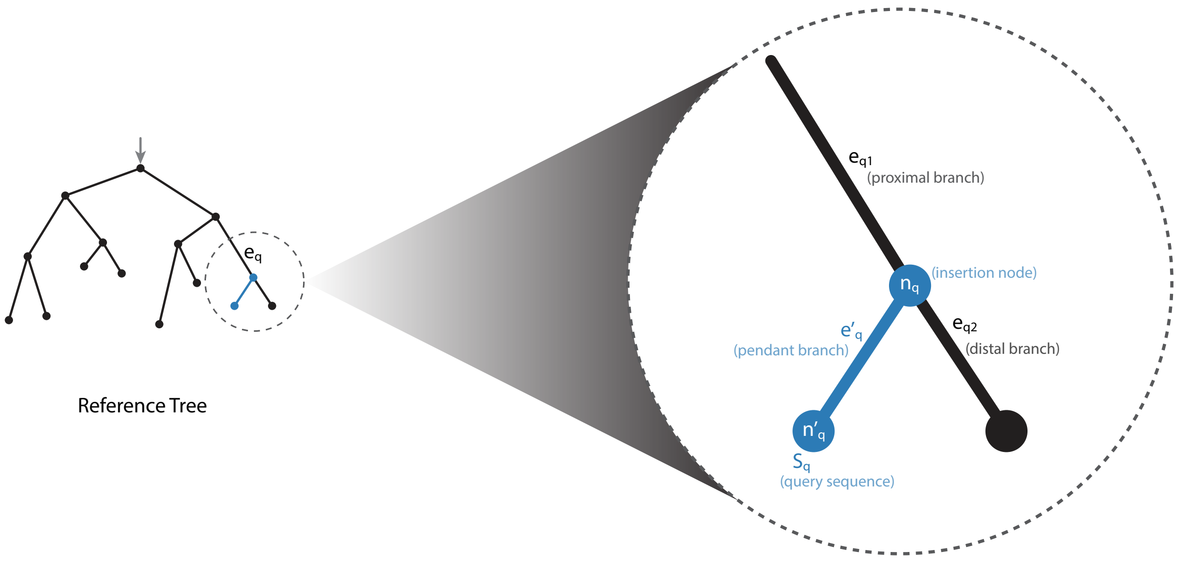


Figure 2.11 – Phylogentic placement of a single query sequence S_q onto a reference tree. Alterations to the reference tree are indicated (blue). An existing branch e_q is chosen such that the query fits well with respect to its evolutionary context. A new node n_q is added that divides e_q into two branches, the proximal and distal parts (e_{q1} and e_{q1}). A new leaf node labeled with S_q is added to $\mathcal{T}_{\text{ref}}$ and connected to n_q via a new branch e'_q, the pendant branch. The resulting tree with all reference sequences and S_q is called $\mathcal{T}_q$. All added nodes and branches (blue) are not permanent changes to $\mathcal{T}_{\text{ref}}$, but rather hypothetical alterations that characterize the phylogenetic relationship between S_q and the references.

are $2 \cdot (m - 3)$ possible locations. At each location an approximate maximum likelihood score is calculated for the resulting tree with $m + 1$ sequences and the query is placed at the position with the maximal score. EPA-NG [15] is a recently released upgrade to EPA that is designed for large collections of data sets by adding multi-core processing and improved heuristics for faster computations. The software RAPPAS [14] is also based on reference sequence alignments, but does not use maximum likelihood to infer placement positions. Instead, RAPPAS calculates contiguous k-mers based on the references that occurred with high probability at inner nodes of the reference tree. These k-mers are inferred from ancestral sequences at the inner nodes—using ancestral sequence reconstruction—and saved in a database. Queries are placed with respect to their k-mers that are also found in this database. On the contrary, APPLES [13] is a distance-based method for PP. It estimates phylogenetic distances between queries and references and specifies the optimal placement position to be the one that minimizes the sum of squared differences between the sequence distances present in the tree and the ones calculated. APPLES does neither necessarily depend on a reference MSA nor on assembled reference sequences. Figure 2.12 provides an overview of the procedure of phylogenetic placement and available algorithms.

PPLACER PPLACER offers a Bayesian and a ML placement mode. In its Bayesian mode, it evaluates the posterior probability of potential placements conditioned on the fixed phylogenetic tree and branch lengths. In its ML-mode, PPLACER calculates likelihood values for each placement position in order to find the branch e_q in $\mathcal{T}_{\text{ref}}$ such that the likelihood of the resulting tree $\mathcal{T}_q$ is maximal. The basis for the ML inference is an existing alignment of each query against all reference sequences. Then, S_q is inserted at all possible $2 \cdot (m-3)$ branches of $\mathcal{T}_{\text{ref}}$. PPLACER normalizes the calculated likelihood values across all branches to one. The normalized likelihood values are referred to as *likelihood weight ratios* (LWRs). Each query is not only placed at a single, but at multiple potential locations in the tree according to and weighted by the LWRs. PPLACER also introduced a measure to quantify placement uncertainty, the *expected distance between placement locations* (EDPL). The EDPL measures the sum of distances between placement locations weighted by their LWR. For two placement positions i and j of the same query S_q their likelihood weight ratios are denoted as LWR_i and LWR_j, the distance between i and j as $d_{ij}(\mathcal{T}_{\text{ref}})$, and the total length of $\mathcal{T}_{\text{ref}}$ as $L(\mathcal{T}_{\text{ref}})$. Then

$$\text{EDPL}(S_q) = \sum_{ij} \frac{\text{LWR}_i \cdot \text{LWR}_j \cdot d_{ij}(\mathcal{T}_{\text{ref}})}{L(\mathcal{T}_{\text{ref}})} \tag{2.29}$$

is the expected distance between the placement locations for a query S_q. A large EDPL indicates high placement uncertainty since the proposed positions of the query sequence occur in spatially distant regions of the reference tree. If the placement of a query sequence is rather uncertain, it may be discarded prior to downstream analysis using the EDPL. PPLACER restricts the ML computation time by applying several heuristics and optimizations during the placement process: Branch lengths are not optimized for all possible placement positions; instead, a search process termed the *baseball* heuristic is used. For this, a conditional maximum likelihood vector is calculated at the midpoint of each edge as explained in Subsec. 2.5.4. Given a query sequence, the edges are then sorted by their potential best fit and examined in this order. Full branch length optimization is successively performed for each edge in the specified order until several edges with low log-likelihood values come up. This greatly reduces the number of edges that need to be examined in detail, resulting in faster runtimes. PPLACER also offers another option to speed up the placement called *friend finding process*. Here, every query is compared to queries that were already placed on the tree. If a query S_q exceeds a specified similarity threshold to a previous query S_p, S_q adopts the placement location from S_p. Otherwise, the branch length optimization for S_q is initialized with the branch lengths that were already chosen for the most similar previous query sequence. Recently, the updated version PPLACERDC has been published, which is optimized for large data sets [330].

EPA and EPA-NG The *evolutionary placement algorithm* (EPA) [10] approaches PP similar to PPLACER. Again, it places a query on multiple branches in $\mathcal{T}_{\text{ref}}$ according to the calculated and normalized likelihood values. As PPLACER, EPA does not calculate the exact likelihood of all possible placement trees but instead provides a fast approximation method as well as a slower but more accurate approach. In both methods, the branch lengths are fixed except for the newly added branch e'_q and the two parts e_{q1} and e_{q2} of the placement edge e_q. The fast method simply sets the lengths of the distal and proximal branches to half the original length of e_q, therefore $l(e_{q1}) = l(e_{q2}) = l(e_q)/2$, and the length of the pendant branch $l(e'_q)$ to a default value. The slow method optimizes the three branch lengths of e_{q1}, e_{q2}, and e'_q with the Newton–Raphson method. Based on the presented experimental results, it is argued that this approximation results only in little loss of accuracy and a more computationally intensive re-calculation of all branch lengths is not required. EPA-NG is a newer version of EPA that comes with additional heuristics to greatly speed up the placement procedure. It also introduces parallelization designed for the placement of large collections of query sequences from metagenomic experiments. The two main improvements are a *preplacement* step and a *masking* heuristic. Evaluating the likelihood at all possible branches of $\mathcal{T}_{\text{ref}}$ is computationally expensive, especially when a large number of references is available. The preplacement step

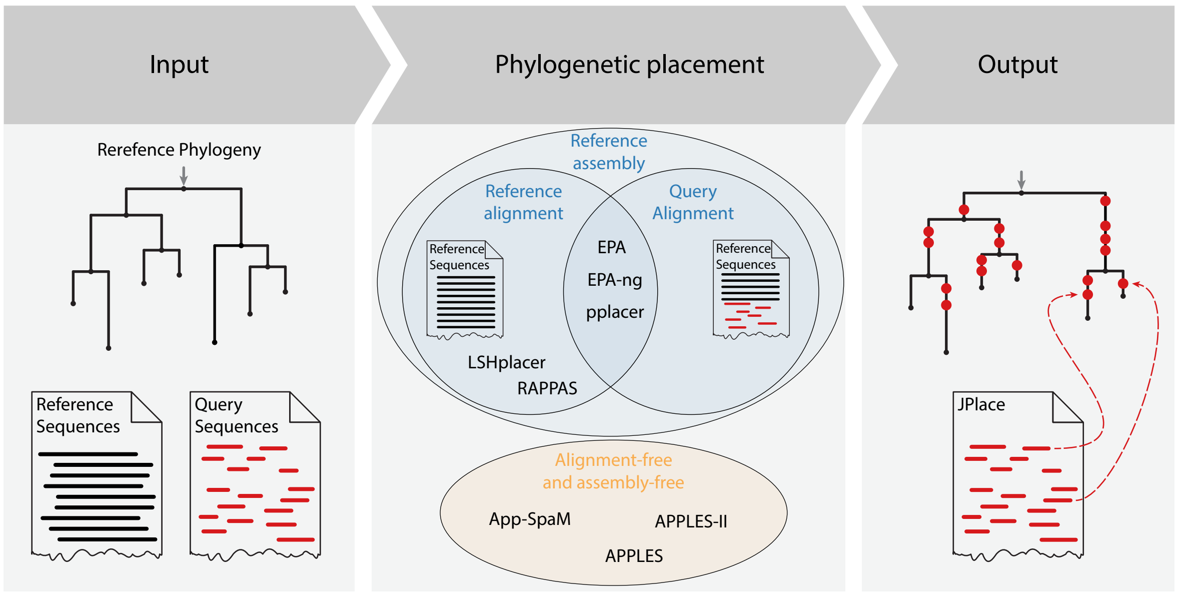


Figure 2.12 – Overview of the phylogenetic placement process and the algorithms currently available. The input consists of reference sequences, a reference phylogeny, and a set of query sequences. Different placement algorithms are grouped according to their requirements for aligned or assembled sequences. The result is a hypothetical position for each query sequence in the reference tree, usually stored in JPlace files. Each entry in the JPlace file corresponds to the placement of a query onto the reference tree.

circumvents this problem by selecting a subset of candidate branches that are likely to yield high likelihood values; an idea that is similar to the baseball heuristic of PPLACER. The thorough placement via ML is then performed only on these candidate branches. The masking heuristic eliminates columns from $\mathcal{A}_{\text{ref}}$, which are determined to have little influence on the likelihood calculations. Additionally, for a query S_q the *core* alignment is defined as only those columns that remain after pruning leading and trailing spaces around S_q when aligned against $\mathcal{A}_{\text{ref}}$. EPA-NG only uses the masked core alignment when calculating the likelihood values for placement positions. EPA-NG achieves comparable accuracy to EPA and PPLACER, however, with greatly accelerated runtimes.

RAPPAS and LSHPlacer The program RAPPAS [14] takes a different approach compared to the algorithms introduced so far: It bases the placement on a pre-computed set of 'phylogenetically informing' k-mers that have a decent probability to occur in query sequences that are related to the reference sequences. These k-mers are termed *phylo-k-mers* and they are calculated based on the reference alignment via *ancestral sequence construction*. The placement process consists of two steps: In the first step, a database of phylo-k-mers is constructed from the reference alignment, the so-called *pkDB*. In the second step, queries are placed on the reference phylogeny based on their k-mers that are also present in the pkDB. Given the reference MSA $\mathcal{A}_{\text{ref}}$ over n sites and their phylogenetic tree $\mathcal{T}_{\text{ref}}$, the pkDB is inferred as follows: At every internal branch b of $\mathcal{T}_{\text{ref}}$ a new hypothetical node h_b is added that divides b evenly into two new branches of identical length. For every node h_b another new leaf node l_b is added with a new branch connecting h_b and l_b. The length of this new branch is chosen as the average length of all paths from h_b to all leaves below. The newly added nodes h_b and l_b are referred to as *ghost* nodes. For each ghost node and each possible k-mer, the probability of observing this k-mer at this node is computed. For this, a matrix $G = \mathbb{R}^{|\Sigma| \times m}$ is created at each ghost node where each element $g_{\sigma,j}$ in G represents the marginal probability of observing the symbol $\sigma \in \Sigma$ at site $j \in [n]$ at the ghost node. Then, the probability to observe a k-mer with the symbols $\sigma_1\sigma_1 \ldots \sigma_k$ starting at position $i \in [n-k+1]$ of the alignment is calculated as

$$P(\sigma_1\sigma_1 \ldots \sigma_k) = \prod_{l=1}^{k} g_{\sigma_l, i+l-1} \, .$$

Each k-mer with a probability above a defined threshold is recorded in the pkDB together with its original branch b and its associated probability. The resulting k-mers do not have to occur in the reference sequences themselves, but the pkDB is rather a collection of those k-mers that have a high similarity to k-mers of the references. Thus, they are likely to occur in unknown queries with a phylogenetic origin at internal branches of $\mathcal{T}_{\text{ref}}$. Thus, while RAPPAS still depends on the reference MSA, it does not depend on the query alignment any more. Given an existing pkDB, a voting procedure determines the placement positions for the queries: For a query S_q, all k-mers from the query are extracted. Every k-mer of S_q is then matched against the pkDB and casts votes according to its entries in the pkDB. A k-mer votes for every internal branch of $\mathcal{T}_{\text{ref}}$ where it occurs, weighted by its occurrence probability as stored in the pkDB. Then, the top weighted internal branches are considered as possible placement locations and recorded in a JPlace file. After constructing the pkDB once, multiple runs with different input query reads can be performed, making RAPPAS a good choice when a fixed set of references is used for several experiments.

The program LSHPlacer [331] has an algorithmic structure similar to RAPPAS. It also performs ancestral sequence reconstruction at the inner nodes of $\mathcal{T}_{\text{ref}}$, resulting in an extended set of reference sequences. To find a placement position for a query sequence S_q *locality sensitive hashing* is utilized: Thus, the idea is to create a set of randomized hash tables that holds the extended set of reference sequences. The hash tables are designed so that hashing S_q will likely result in a collision with those (ancestral) reference sequences that exhibit a large similarity with S_q. From this, a single, possibly ancestral, reference S_r is determined that has the highest similarity to S_q; the query is then inserted into the tree at a branch near the node associated with S_r using a local search procedure. As for RAPPAS, the hash tables only need to be constructed once and then allow repeated usage with different queries.

APPLES and APPLES-II Another software for PP is the *Accurate Phylogenetic Placement using LEast Squares* program, called APPLES [13]. In contrast to pplacer and EPA, APPLES is a distance-based method that does not require aligned reference sequences. While other programs focus primarily on short genes and marker regions, APPLES also considers long references that do not necessarily originate from homologous regions. Additionally, those sequences do not have to be assembled; it is sufficient to supply a set of reads for each reference or merely a distance matrix that quantifies the distances between all query-reference pairs. This flexibility is achieved by relying solely on distance estimates between query and reference sequences to calculate placement positions: Similarly to least-squares phylogeny reconstruction, APPLES performs a least-squares optimization that minimizes the difference between calculated distances for a query S_q to all references and the distances given in the placement tree $\mathcal{T}_q$. For a query S_q and reference sequences $S_i, i \in [m]$, the distance between S_q and each reference is designated as $\delta_{iq}, i \in [m]$. Then, APPLES solves

$$\arg\min_{\mathcal{T}_q} \sum_{i=1}^{m} w_{iq} \left(\delta_{iq} - d_{iq}\left(\mathcal{T}_q\right)\right)^2 \tag{2.30}$$

whereas $d_{iq}(\mathcal{T}_q)$ is the distance between S_i and S_q in $\mathcal{T}_q$. For this, S_q has to be inserted at every possible branch of $\mathcal{T}_{\text{ref}}$ and branch lengths are adapted. Again, as discussed in Subsec. 2.5.2, different values are plausible for the weights w_{iq}; for example, δ_{iq}^{-2} is used as default in APPLES-II to reduce the effect of large distances on the estimation process. APPLES uses dynamic programming to solve Eq. 2.30 in linear time. Alternatively to Eq. 2.30, APPLES provides the possibility to produce a placement tree that satisfies the minimum evolution principle. The more recent version APPLES-II [332] adds a divide-and-conquer search strategy for the placement of queries. This addition allows APPLES-II to process data sets that comprise thousands of references, as promising placement regions are quickly located within the reference tree.

2.7.2 Further Considerations

The presented algorithms constitute a common framework to perform PP and provide results in the standardized JPlace format [329]. However, the preparation of appropriate input data requires more detailed considerations. Subsection 2.6.3 already discussed the importance of reference databases for the task of read assignment, including their size, completeness, and correctness, in order to derive taxonomic labels of high accuracy. Similarly, methods for PP are highly dependent on the reference alignment $\mathcal{A}_{\text{ref}}$ and the reference tree $\mathcal{T}_{\text{ref}}$. Besides the number of reference sequences, the means by which they were chosen from available databases with respect to the anticipated queries is of relevance. Optimally, the references are sampled equally among those taxonomic groups that are expected to appear in the query sequences. In case of doubt, the sample should comprise more taxonomic groups, as this provides the possibility to perform placements at deep inner branches of the reference tree. Nevertheless, a dense sampling of multiple sequences from closely related organisms should be avoided: If multiple similar organisms are present in the references, for example, multiple strains of the same species, query placements belonging to a single organism are distributed among multiple branches of $\mathcal{T}_{\text{ref}}$ which limits the specificity of the placements and skews subsequent analysis steps. Efforts have been made to automatically infer appropriate sets of references from large databases or to perform placement in 'nested' phylogenies to overcome these issues [333]. Still, it is argued that manual curation of reference sequences remains indispensable [334] and, thus, the selection of an adequate set of references continues to be a labor intensive bottleneck for placement studies.

Most PP methods require an alignment $\mathcal{A}_{\text{ref}}$ of the references as input to derive ML values or phylo-k-mers. It is important that $\mathcal{A}_{\text{ref}}$ is of high quality and that the same evolutionary model is used for the construction of $\mathcal{A}_{\text{ref}}$ as for the ML-based placement programs. In most cases, $\mathcal{T}_{\text{ref}}$ is also reconstructed from $\mathcal{A}_{\text{ref}}$ using ML phylogeny reconstruction. The need for alignments restricts the application of PP programs to assembled sequences of limited length. As a consequence, the most common use case of PP has been metataxonomics. One advantage of this area of application is the existence of large annotated databases from which references can be chosen effortlessly; see Sect. 2.6.2. Sometimes, database entries are also associated with existing phylogenetic trees; in the optimal case, those trees are supplied by the respective research communities that focus on the organisms under consideration. PP then performs a phylogenetic classification of the unidentified query sequences from upcoming metataxonomic experiments. In contrast, performing alignment-based PP in *metagenomics* is more difficult: it requires that appropriate sequences of all reference organisms have been assembled prior to constructing the reference alignment. The references must span the same genomic regions which comprise, in many cases, the whole genome. In addition to $\mathcal{A}_{\text{ref}}$, multiple programs for PP also require an alignment of the query sequences against $\mathcal{A}_{\text{ref}}$. It is beneficial to use phylogenetic information to improve the query alignment. For this, several *phylogeny-aware* alignment tools exist that align short reads to existing MSAs [335, 336]. Other tools, such as MAFFT [87], provide specific options to align short reads against longer alignments without altering the existing alignments. This saves significant computational time and gives better results than using regular multiple sequence aligners [336]. While several placement algorithms place a single query at multiple branches of $\mathcal{T}_{\text{ref}}$, it is often sufficient or recommended to specify only a single placement position, for example, when visualizing placements of many queries or when comparing the diversity between samples. Multiple specified placement positions are usually *weighted*, for example, by likelihood-weight ratios inferred from ML methods. If the specification of a single placement branch is required, a *consensus placement* can be inferred from the multiple weighted placements. The most straightforward strategy is to simply choose the highest weighted placement branch within $\mathcal{T}_{\text{ref}}$ as consensus placement.

Besides algorithms that perform phylogenetic placement, a variety of ancillary and affiliated tools exist that focus on the pre- or post-processing of PP data. The C++ libraries GENESIS and GAPPA offer broad functionality to operate on placement data and JPlace files efficiently. In addition to basic input-, output-, and parsing-functions, they also implement a variety of sophisticated methods to analyze metagenomic samples by means of their placement data; for example, they provide PP-based distance measures between samples, several PP-based clustering algorithms, and the so-called *Edge-PCA*, a principal component analysis that operates on a feature space formed by the placement weights scattered across the branches of $\mathcal{T}_{\text{ref}}$. SCRAPP [337] is another advanced analysis method based on phylogenetic placement data. SCRAPP calculates a phylogeny-aware measure of the within-sample diversity—also referred to as α-diversity—based on the distribution of read placements across the branches of $\mathcal{T}_{\text{ref}}$. For this, SCRAPP infers new phylogenetic trees for all those query sequences that are placed on the same branch in $\mathcal{T}_{\text{ref}}$; this constitutes a novel approach and the resulting trees are termed 'branch query phylogenies'. Another recently proposed use case for phylogenetic placement is the rooting of phylogenetic trees [334]. For this, sequences of an outgroup are placed on a tree and resulting placement branches indicate where the root may reside. Likewise, many other algorithms utilize the output of phylogenetic placement programs for purposes other than metagenomic read identification or within- and between-sample comparisons. Two prominent examples are CHECKM [338] and PICRUST2 [339]. CHECKM estimates the completeness and contamination of genomes by incorporating reference sequences from the evolutionary neighborhood of query sequences. PICRUST2 predicts the functions of metagenomic communities by incorporating PP data. There is an excellent review that discusses all of these aspects in great detail [334].

2.7.3 Evaluating Phylogenetic Placement Algorithms

The evaluation of phylogenetic placement algorithms is far from trivial. In a real-world scenario, the prerequisites for performing PP are a set of reference sequences, a reference phylogeny, and a set of query reads. Here, the reference phylogeny $\mathcal{T}_{\text{ref}}$ refers to the *inferred* phylogenetic tree estimated from the reference sequences with any feasible method for phylogenetic tree inference. In contrast, the *true underlying* phylogenetic tree is referred to as $\mathcal{T}^*$. This tree is generally unknown when working with real-world data. Optimally, $\mathcal{T}_{\text{ref}}$ coincides with $\mathcal{T}^*$ but in practice the inferred tree is usually different. By the definition of PP, the correct—or 'best'—placement position of the query reads in $\mathcal{T}_{\text{ref}}$ is also unknown. Additionally, it is challenging to specify which position qualifies as the 'best' position in the inferred phylogenetic tree in the first place; even with knowledge about its true position within $\mathcal{T}^*$. As an example, lets assume that a query S_q belongs to a species that branched off at an internal branch b^* of the true underlying phylogeny $\mathcal{T}^*$ (the species itself is not contained in $\mathcal{T}^*$). The branch b^* divides all nodes of $\mathcal{T}^*$ into two distinct groups; such a bipartite grouping of nodes is a *split* of the tree. Each branch induces a distinct split of nodes within the tree. However, the split induced by b^* might not be present in the inferred phylogeny $\mathcal{T}_{\text{ref}}$. Or, in other words, there is not always a branch b in $\mathcal{T}_{\text{ref}}$ that induces the same split as b^* in $\mathcal{T}^*$. Thus, it is impossible to determine the 'correct' placement position with respect to $\mathcal{T}^*$ in most cases. Instead, a branch b from $\mathcal{T}_{\text{ref}}$ has to be determined whose split is most similar to the split of nodes that b^* induces in $\mathcal{T}^*$. And even if the split induced by b^* is also present in $\mathcal{T}_{\text{ref}}$, it is unknown which of the branches in $\mathcal{T}_{\text{ref}}$ actually corresponds to b^* in a real-world scenario. Three different approaches are considered to solve this dilemma in practice. Each of them represents a proxy to measure the quality of inferred placement positions with respect to the true placement position in $\mathcal{T}^*$.

The first approach operates directly on the supplied real-world data by comparing the congruence between placement positions inferred from several placement programs. This idea is termed *evaluation by reconciliation*. Each query S_q is placed on $\mathcal{T}_{\text{ref}}$ with a set of chosen programs. The result is a collection of weighted placement positions for each program and the congruence among the results provides information about the accuracy of the programs. For simplicity, only the consensus placement for each query and for each program is considered, respectively; however, the concept can be easily extended to incorporate a set of weighted placement positions for each program. Intuitively, if one placement program $\mathcal{P}$ calculates a placement location that substantially deviates from the placement locations of the other programs, it may be considered less accurate. In contrast, if the position determined by $\mathcal{P}$ corresponds to the positions of the other programs, it may be considered a reliable placement. This approach expects that the involved placement programs have either no placement biases or different biases that balance each other out. Thus, the reconciliation process assumes that the consensus placement of several programs is an unbiased estimate of the most appropriate placement position in $\mathcal{T}_{\text{ref}}$; while this assumption is difficult to prove, the reconciliation still constitutes a reasonable basis to evaluate the divergence of placement estimates when using different approaches to PP. Furthermore, it highlights potential differences in PP programs and is a useful tool to assess advantages and drawbacks of the programs when used in conjunction with other evaluation methods. The reconciliation process is especially appropriate when a subset of the participating programs has already undergone other external accuracy evaluations, see below. Then, the placement positions of those programs serve as a baseline for the placement positions of new programs. For example, EPA-NG performed an accuracy evaluation through reconciliation by comparing its placements with those of PPLACER and EPA. To compare the likeness of query placements between programs, the *phylogenetic Kantorovich-Rubinstein* (PKR) metric was used: Given two weighted distributions of placement positions across branches of $\mathcal{T}_{\text{ref}}$, PKR is defined as the minimal amount of work required to transfer one distribution to the other one. *Work* is defined as the weight of a placement position multiplied by the distance it has to travel in the phylogenetic tree measured by the number of nodes. For two programs, the minimum amount of work is averaged over all corresponding query placement distributions to obtain a single measure for their placement similarity. Although reconciliation procedures are easy to perform, they do not consider the true tree $\mathcal{T}^*$ in any way. Furthermore, they cannot ultimately discriminate which of the involved programs performs better or worse.

The second approach contrasts the first one and was first proposed in APPLES [13]: Instead of using real-world data to evaluate the accuracy, the data is simulated from scratch in order to have full information about the correct placement locations. First, the true reference tree $\mathcal{T}^*$ is simulated for m species. Then, according to $\mathcal{T}^*$, the sequences of the m species at the leaves are simulated, resulting in the true reference alignment $\mathcal{A}^*$. Estimating the placement accuracy directly on $\mathcal{T}^*$ and $\mathcal{A}^*$ is unrealistic as neither of them contain errors. To better replicate the real-world scenario, the unaligned sequences of $\mathcal{A}^*$ are realigned to create an inferred MSA $\mathcal{A}_{\text{ref}}$, which is used to calculate an inferred tree $\mathcal{T}_{\text{ref}}$. Thus, both $\mathcal{A}_{\text{ref}}$ and $\mathcal{T}_{\text{ref}}$ now may contain errors. Query sequences are derived by a *leave-one-out* procedure: A single sequence S_q is removed from $\mathcal{A}_{\text{ref}}$ and $\mathcal{T}_{\text{ref}}$. The resulting pruned tree is called $\mathcal{T}_{\text{ref}|S_q}$, whereas $\mathcal{T}^*_{|S_q}$ is the true tree without S_q. S_q is then placed back onto $\mathcal{T}_{\text{ref}|S_q}$ resulting in the placement tree $\mathcal{T}_q$. The placement of S_q in $\mathcal{T}_q$ can now either be compared to its position in $\mathcal{T}_{\text{ref}}$ or to its original position in $\mathcal{T}^*$. However, both cases do not account for the discrepancies between the true and inferred tree. Instead, APPLES uses a metric called *delta error*. Let $B(\mathcal{T})$ be the set of all splits of $\mathcal{T}$ induced by its branches. The delta error is defined as

$$\Delta e(\mathcal{T}_q) = |B(\mathcal{T}^*) \setminus B(\mathcal{T}_q)| - |B(\mathcal{T}^*_{|S_q}) \setminus B(\mathcal{T}_{\text{ref}|S_q})| \, .$$

Thus, the delta error measures the difference in the number of splits between $\mathcal{T}_q$ and $\mathcal{T}^*$ that are solely introduced by the placement of S_q and that do not originate from other discrepancies between the true tree and the inferred tree. The whole procedure is performed for multiple different query sequences taken from $\mathcal{A}_{\text{ref}}$ and the delta error is averaged over all repetitions to measure the overall accuracy of the software under evaluation. If $\mathcal{T}_{\text{ref}}$ and $\mathcal{T}^*$ are different the delta error is greater than zero even when S_q is placed at the exact location of $\mathcal{T}_{\text{ref}}$ where it was pruned from. To obtain a low delta error, S_q must be placed optimally with respect to $\mathcal{T}^*$. Thus, the delta error assumes that the phylogenetic placement of S_q should result in an improved position of the species compared to its position in the reconstructed $\mathcal{T}_{\text{ref}}$. However, both the placement and the tree reconstruction are based on the same data—$\mathcal{A}_{\text{ref}}$—and, hence, it is questionable to assume that placement algorithms achieve a higher accuracy with respect to the true underlying tree than phylogenetic reconstruction algorithms. If this would be the case, phylogenetic reconstruction should be skipped altogether and trees should be inferred iteratively via placement to achieve the highest accuracy, see also Sec. 5.1. Although there might be rare instances where placement indeed provides a better resolution of a species placement, it is improbable that placement generally achieves a higher accuracy than phylogenetic reconstruction.

The third approach to evaluating PP programs uses real-world data for $\mathcal{T}_{\text{ref}}$ and $\mathcal{A}_{\text{ref}}$ together with artificially created query sequences. The most common course of action, first conducted in PPLACER and EPA [10, 12], follows a pruning-based procedure: In one *pruning event* a single reference sequence S_q is pruned from $\mathcal{T}_{\text{ref}}$ resulting in the pruned tree $\mathcal{T}_{\text{ref}|S_q}$. The internal branch where S_q previously branched off is called b_s. The same sequence S_q is also removed from $\mathcal{A}_{\text{ref}}$ resulting in $\mathcal{A}_{\text{ref}|S_q}$. Then, the branch lengths of $\mathcal{T}_{\text{ref}|S_q}$ are re-optimized and S_q is placed back onto the pruned reference tree resulting in $\mathcal{T}_q$. The original position of S_q in $\mathcal{T}_{\text{ref}|S_q}$ is known, namely the branch b_s. This branch is considered to be the correct placement branch for the query sequence. The proposed placement position of a PP program is compared to this correct position. The closer the query is placed to b_s, the higher the accuracy of the placement algorithm. Again, it is not taken into account that $\mathcal{T}_{\text{ref}}$ might differ from the true topology $\mathcal{T}^*$ as the true topology is unknown anyway. Let b_q be the proposed placement branch of S_q. There are different metrics that measure the accuracy of the proposed placement branch b_q with respect to the correct placement branch b_s. The

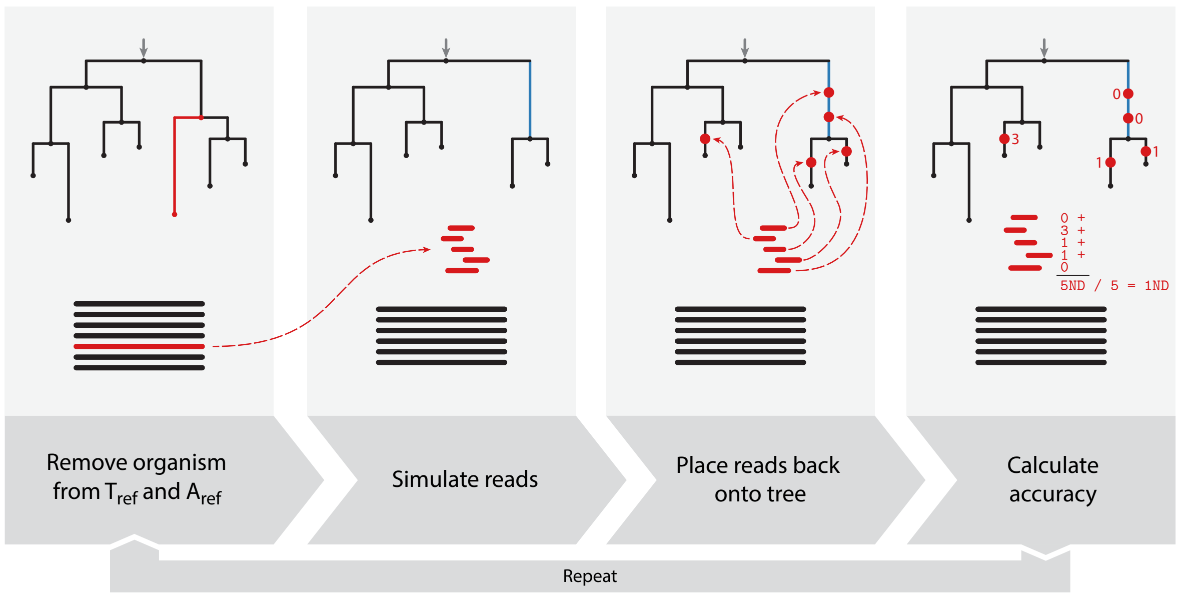

0
0
3
1
1
0 +
3 +
1 +
1 +
0
5ND / 5 = 1ND
Remove organism from T_{ref} and A_{ref}
Simulate reads
Place reads back onto tree
Calculate accuracy
Repeat

Figure 2.13 – Pruning-based accuracy evaluation procedure for phylogenetic placement algorithms: A data set consisting of a reference tree and reference sequences is provided. A random reference is chosen and removed from the tree (red node and branch) and from the reference alignment (red sequence). The location where the reference was pruned is indicated in the tree (blue branch). Then, reads are simulated from the removed sequence and placed back onto the pruned tree. The accuracy is measured as the average number of nodes (ND) between the proposed placement position and the true position. The procedure is repeated multiple times. In PEWO, instead of removing a single sequence, a random node is chosen and all sequences below are pruned.

simplest metric, called *node distance*, counts the number of nodes that separate the two branches b_s and b_q. If the query is placed exactly at the location where S_q was pruned, the accuracy is optimal with a node distance of 0. Alternatively, branch lengths may be included in the metric by calculating the sum of all branch lengths between the proposed and correct placement positions. When multiple proposed placement positions for a single query exist, the weighted average node distance over all proposed locations is called the *expected node distance* [340]. Multiple pruning events are performed consecutively, and the average of all pruning events serves as an overall accuracy measure for an algorithm; see Fig. 2.13.

Variations of the pruning-based procedure differ mainly with respect to the generation of query sequences. Instead of placing the whole pruned sequence S_q it is also possible to simulate short reads from S_q to mimic a metagenomic scenario. As previously, the simulated reads are then placed onto the pruned reference tree and their average accuracy with respect to the correct branch is calculated. Query reads may be simulated in different ways; for example, by simply splitting S_q into reads of a specified length. Other solutions include the sampling of reads from a uniform distribution over the sequence positions of S_q until a specified coverage is reached, or the simulation of reads with a specified error profile according to common sequencing platforms. For the latter approach, a variety of tools exist: ART [341] simulates reads with different characteristics derived from large sequencing platforms. MetaSim [342] generates metagenomic reads from a whole genome database to simulate common mixed metagenomic datasets, and NanoSim [343] simulates long reads from recent sequencing platforms such as Oxford Nanopore Technologies [278]. The pruning-based accuracy evaluation described so far only prunes a single reference sequence in each pruning event. However, this results in rather homogeneous pruning events when $\mathcal{T}_{\mathrm{ref}}$ is an evenly sampled reference tree. To better mimic real-world scenarios where query sequences might have no closely related relatives in $\mathcal{T}_{\mathrm{ref}}$—or originate from deep inner nodes—it is advisable to generate more diverse pruning events. One solution is to alter the number of pruned reference sequences in each pruning event, for example, by pruning random subtrees from $\mathcal{T}_{\mathrm{ref}}$ instead of only single sequences. Then, all removed sequences are placed back onto the pruned tree.

The *Placement Evaluation WOrkflows* (PEWO) [340] is a software for the evaluation of PP algorithms with respect to their accuracy and speed. PEWO offers three different evaluation workflows, namely the pruning-based accuracy evaluation (PAC), a likelihood-based accuracy evaluation (LAC), and the resources workflow (RES). The PAC is identical to the third evaluation procedure described above with the following specifications: In each pruning event, a randomly selected node in $\mathcal{T}_{\mathrm{ref}}$ is chosen and all references below that node are pruned. Then, query reads of a specified length are simulated by splitting the pruned sequences in segments of the specified length and every read is placed back onto the tree. The average accuracy over all query placements is measured by the node distance and expected node distance metrics. The pruning process is repeated multiple times with different subtrees and the average accuracy over all pruning events constitutes the overall performance.

Chapter 3

Alignment-free Phylogenetic Placement with App-SpaM

> The content of this chapter is derived from the peer-reviewed open-access publication
>
> Matthias Blanke and Burkhard Morgenstern.
> "App-SpaM: Phylogenetic placement of short reads without sequence alignment."
> In: *Bioinformatics Advances* (2021).
>
> All text has been thoroughly adapted and new content has been added by Matthias Blanke to fit this thesis. All figures are original works created by Matthias Blanke.

Phylogenetic placement (PP) is a unique approach to characterizing biological sequences that interconnects the areas of metagenomics, metataxonomics, and phylogenetics. Most programs that perform PP are dependent on alignments and do not scale well with the amount and diversity of sequencing data made available by recent sequencing technologies. Until now, PP programs were mainly used to assess the composition and abundance of species in samples from metataxonomic experiments by placing a set of query reads into comprehensive reference phylogenies. Alignment-free programs provide an unparalleled opportunity to promote the field of PP as they are capable of handling large amounts of diverse sequencing data; in addition, they are applicable in challenging circumstances, for example, when sequences are not or only partially assembled, or when sequences are under the influence of large-scale evolutionary events. We believe that an alignment-free program for general use PP is a useful addition to the programs and libraries for PP currently available. Furthermore, it opens the door to novel use cases in addition to the identification of metataxonomic samples, which we discuss in more detail in Chpt. 5.

One method for alignment-free distance estimation is the spaced-word matches approach that finds non-exact gap-free micro-alignments to estimate the average number of nucleotide substitutions between two sequences. Its implementation FSWM and variants thereof produce distance estimates on par with other state-of-the-art alignment-free programs [344]. Here, we present the ***A****lignment-free* ***p****hylogenetic* ***p****lacement algorithm based on* ***Spa****ced-word* ***M****atches* (App-SpaM). App-SpaM employs filtered spaced-word matches to quickly calculate query-reference distances and performs phylogenetic placement using fast heuristics.

3.1 Alignment-free Phylogenetic Placement Based on Spaced-Word Matches

As for other placement programs, the input to App-SpaM consists of m reference sequences, an edge-weighted phylogenetic tree $\mathcal{T}_{\text{ref}}$ comprising the m references, and a set of query sequences to be placed onto $\mathcal{T}_{\text{ref}}$. Unlike other programs, App-SpaM does not require that reference sequences are aligned or assembled. Instead, we use a distance-based strategy comprising three consecutive steps: First, we calculate filtered spaced-word matches between all queries and references; see Subsec. 2.4.3 for an introduction to this technique. Second, we filter out non-homologous matches, count the remaining ones between each query and reference, and use them to estimate query-reference distances. Third, a placement position is specified for each query using fast heuristics, and the results are saved in the JPlace format. We performed a comprehensive evaluation of App-SpaM and demonstrated that its proposed query placements are of high quality and on par with ML-based approaches when applied to metataxonomic data. Furthermore, App-SpaM requires orders of magnitude less computation time and has a low memory consumption compared to other programs. These results are valid for a wide range of program parameters, such as the number of patterns, their length, and weight. We also applied App-SpaM to unassembled reference sequences as a proof of concept; the placement accuracy remains stable for a high coverage of the references and declines with lower coverages as expected. We tested multiple algorithmic variations of our program, including a sampling procedure for spaced words and the use of different evolutionary models, and present detailed results of their advantages and drawbacks.

3.1.1 Algorithmic Methodology and Evaluation Setup

We first introduce the algorithmic procedure of App-SpaM including certain technical aspects that affected its design choices, followed by a detailed description of the performed evaluation. As the first step, App-SpaM extracts all spaced words from the input sequences with respect to a predefined binary pattern P. It is important to consider the structure of P and, in particular, its length l and the number of match positions. Spaced-word matches are detected based on identical match positions; a lower number of match positions (also called the pattern weight w) entails a larger number of potential matches that are evaluated in the filtering procedure. On the contrary, using a higher pattern weight results in fewer spaced-word matches; this reduces computation time, but homologous regions might be missed in exchange. This trade-off is comparable to the choice of k when using approaches based on continuous words. Generally, using a lower pattern weight is necessary for more dissimilar sequences. The first software program FSWM that employed filtered spaced-word matches was designed to calculate distances between assembled whole genomes. It uses a default pattern with a weight of $w = 12$ and $l = 112$ don't care positions since those parameters yielded consistent results across several tested data sets. However, these parameters were tailored to the task of comparing two whole genomes and do not necessarily work for other types of sequencing data. For example, the program Prot-SpaM uses a weight of $w = 6$ and a pattern length of $l = 40$. This adjustment was carried out to meet the requirements of protein sequences, since they are defined over a larger alphabet ($|\Sigma| = 20$) and evolve at slower rates than DNA sequences. Accordingly, the weight and length of the pattern were also adjusted in App-SpaM: By default, we use a single pattern created by the software rasbhari [165] with length $l = 44$ and weight $w = 12$. The choice of shorter word lengths is motivated by the nature of the query sequences: Especially in applications such as metataxonomics, short query reads are prevalent. A shorter pattern entails substantially more spaced words that arise from a single read. For example, a single read with a length of 150 bp produces 39 spaced words when using $l = 112$ as in FSWM; however, when the pattern length is reduced to $l = 44$ a total of 107 spaced words emerge. This constitutes an increase in the number of spaced words of 274%. This effect becomes even stronger when read lengths are shorter than 150 bp. Another main motivation to choose such short patterns is the computational benefit that is gained, see below.

We term the string of match positions of a spaced word as its *key* K. We determine all spaced words with respect to P for all reference sequences and save them in a list L_r ordered by their key. Accordingly, another ordered list L_q is created for the spaced words of all query sequences. For each spaced word in L_r and L_q we also store the input sequence it originated from and its location within this sequence. Then both lists are divided into blocks consisting of identical spaced words with respect to their key. The ordered blocks of L_r and L_q are traversed simultaneously such that both blocks corresponding to a single key K are selected at once. Each pair of spaced words with one word from each selected block constitutes a spaced-word match with identical match positions. Thus, each pair represents a spaced-word match of the spaced word W with key K between two spaced-word occurrences (W, i) in a query sequence S_q at position i and (W, j) in a reference sequence S_r at position j. For each such spaced-word match, its score is calculated as the sum of all substitution scores of the nucleotides at their don't care positions according to the substitution matrix *HOXD70* [345]. Spaced-word matches with a score below the predefined threshold t are removed; on default, t is set to zero as in FSWM. This filtering procedure ensures that only matches from homologous sequence fragments remain. Matches that occurred purely by chance are filtered out and are not included in any subsequent analysis steps. For each query S_q and reference S_r we save the number of spaced-word matches $s(S_q, S_r)$ with score above t. In addition, we calculate the phylogenetic distance $d(S_q, S_r)$ between each query and reference. For this, we count the average number of nucleotide substitutions $p(S_q, S_r)$ at the don't care positions of the spaced words and correct it using the Jukes-Cantor formula [346]:

$$d(S_q, S_r) = -\frac{3}{4} \ln \left(1 - \frac{4}{3} \cdot p(S_q, S_r)\right) . \tag{3.1}$$

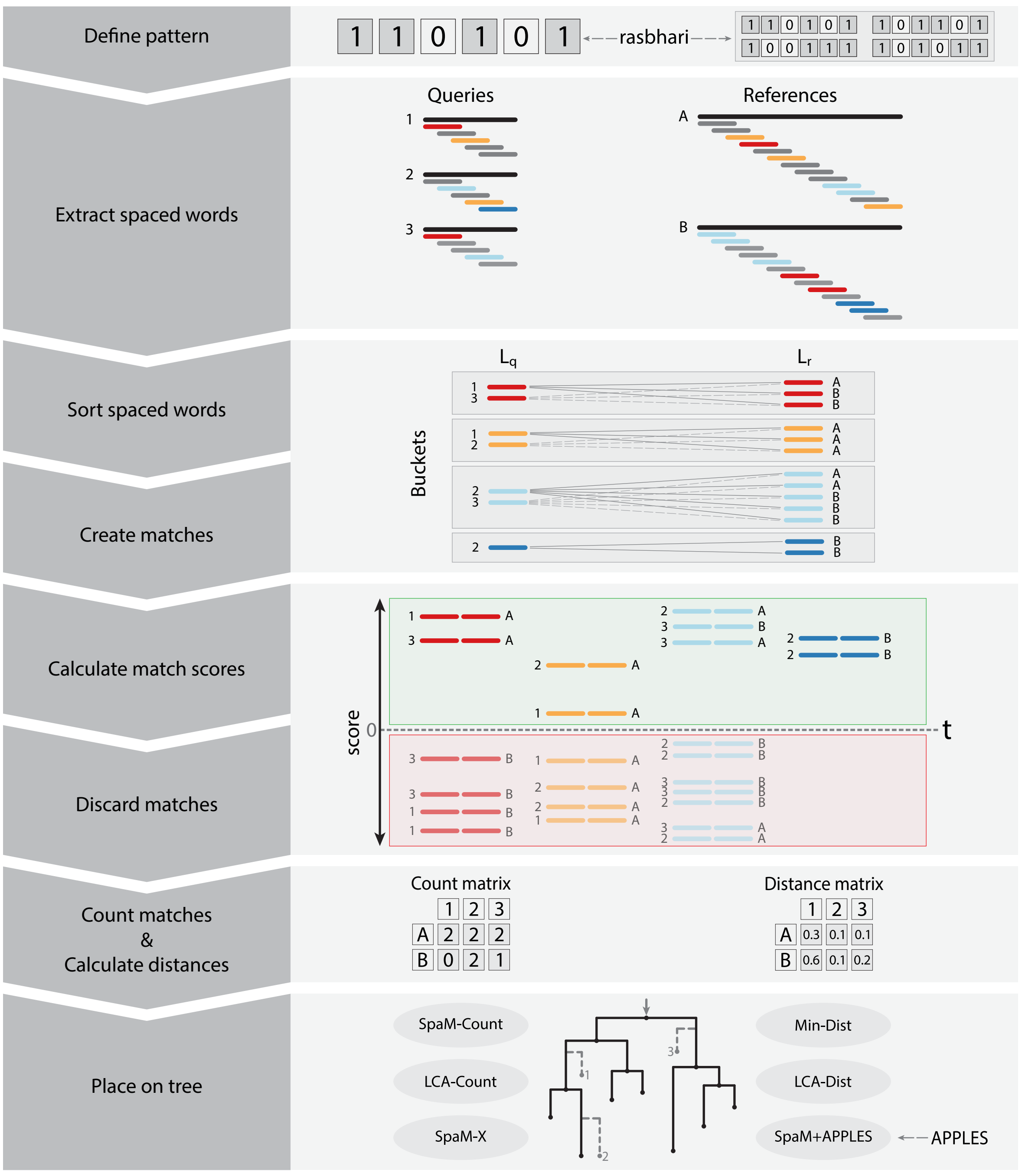
After both lists L_r and L_q are fully traversed, $s(\cdot\,, \cdot)$ and $d(\cdot\,, \cdot)$ are utilized to infer placement positions in $\mathcal{T}_{\text{ref}}$.

We propose and evaluate five heuristics to place a query S_q on $\mathcal{T}_{\text{ref}}$ based on its phylogenetic distances $d(S_q, S_r)$, $r \in [m]$ and the numbers of spaced-word matches $s(S_q, S_r)$, $r \in [m]$. Each heuristic chooses a branch e_q in $\mathcal{T}_{\text{ref}}$ where S_q is placed. As explained in Subsec. 2.7.1, e_q is split into a proximal part e_{q1} and distal part e_{q2} by the newly added insertion node n_q. A new pendant branch e'_q is added that connects n_q with the new leaf node n'_q which is annotated with the query sequence S_q. Subsequently, each heuristic in App-SpaM specifies the length of the newly added distal branch $l(e_{q2})$ and the pendant branch $l(e'_q)$. The length of the proximal branch is always defined as $l(e_{q1}) = l(e_q) - l(e_{q2})$. The heuristics select e_q and calculate branch lengths for the placement of a query sequence as follows:

Min-Dist — This heuristic is a vast simplification of the complex placement process and serves as a basic benchmark. Min-Dist selects the reference S_r that minimizes the distance $d(S_q, S_r)$ across all reference sequences:

$$\underset{S_r,\, r\in[m]}{\arg\min}\; d(S_q, S_r)\,. \tag{3.2}$$

The placement edge e_q is defined as the single edge in $\mathcal{T}_{\text{ref}}$ that is adjacent to the leaf labeled with S_r. In the rare case that multiple references have the same smallest distance to S_q, one of them is chosen at random. We distinguish two situations to infer branch lengths: When $d(S_q, S_r)/2 < l(e_q)$ the lengths of e_{q2} and e'_q are set to $l(e'_q) = l(e_{q2}) = d(S_q, S_r)/2$. Choosing the lengths of e'_q and e_{q2} to be identical is comparable to the assumption of an ultrametric



	1	2	3
A	2	2	2
B	0	2	1

	1	2	3
A	0.3	0.1	0.1
B	0.6	0.1	0.2

Figure 3.1 – The process of App-SpaM: First, a binary pattern is defined using the software rasbhari. Alternatively, multiple patterns may be used. The queries are enumerated numerically (1, 2, 3), and the references alphabetically (A, B). Spaced words (short fragments) are extracted from the queries and references. Spaced words with the same key are highlighted (colored fragments), in contrast to spaced words with singular keys (gray fragments). The lists L_r and L_q are created, sorted by their key, and buckets (gray boxes) with identical key K are delimited (for clarity, words with singular keys are omitted in the depiction). For each bucket, scores between all spaced-word matches with one spaced word from L_r and another one from L_q are calculated. Spaced-word matches with a score below the predefined threshold t are discarded. The remaining spaced-word matches are used to calculate the number $s(S_q, S_r)$ of matches and the phylogenetic distance $d(S_q, S_r)$ between each query S_q and each reference S_r. One of the six proposed placement heuristics is used to place each query onto $\mathcal{T}_{\text{ref}}$.

tree in UPGMA. This case represents a scenario where it is rather reasonable to place S_q on the branch directly above its next similar reference as $d(S_q, S_r)/2$ is smaller than the distance between S_r and its parental node in $\mathcal{T}_{\text{ref}}$. Under the assumption of an ultrametric tree, the suggested placement of S_q is natural. However, if $d(S_q, S_r)/2 \geq l(e_q)$, we set $l(e_{q2}) = l(e_q)$, and $l(e'_q) = d(S_q, S_r) - l(e_q)$. This creates an implicit trifurcation at the node above n'_q as the length of the proximal branch e_{q1} is set to zero. In this second case, the distance between S_q and the most similar reference S_r is greater than the distance between S_r and its parental node. The placement of S_q at e_q is only reasonable if S_q has evolved more rapidly from the parental node of S_r than S_r itself. Otherwise, a placement position of S_q at inner branches of the tree explains the large distance $d(S_q, S_r)$ better.

SpaM-Count — Like Min-Dist, this heuristic places queries only on branches directly above the leaves. But instead of selecting the reference sequence S_r that minimizes the distance to S_q, SpaM-Count selects the branch above the reference that satisfies

$$\underset{S_r,\, r \in [m]}{\arg\max}\; s(S_q, S_r)\,. \tag{3.3}$$

The chosen sequence maximizes the number of spaced-word matches to S_q with a score larger than t. The distance calculation of branch lengths is identical to that in Min-Dist.

LCA-Dist — This heuristic is in direct contrast to the previous two: it identifies the *two* reference sequences S_{r_1} and S_{r_2} with the lowest distances $d(S_q, S_{r_1})$ and $d(S_q, S_{r_2})$. While queries are placed exclusively on branches directly above the leaves with Min-Dist and SpaM-Count, placements with LCA-Dist are placed solely above inner nodes of $\mathcal{T}_{\text{ref}}$. Let n_{lca} be the *lowest common ancestor* of the two leaves labeled with S_{r_1} and S_{r_2} in $\mathcal{T}_{\text{ref}}$. The edge in $\mathcal{T}_{\text{ref}}$ that connects n_{lca} with its parental node is chosen as the placement edge e_q. Let $l(S_{r_1}, n_{\text{lca}})$ be the sum of edge lengths between S_{r_1} and n_{lca}, and accordingly $l(S_{r_2}, n_{\text{lca}})$ the sum of edge lengths between S_{r_2} and n_{lca}. We define

$$\hat{d}(S_q) = \frac{d(S_q, S_{r_1}) + d(S_q, S_{r_2})}{2} \tag{3.4}$$

as the average estimated distance between S_q and the two chosen references. Furthermore, we define

$$\hat{d}(n_{\text{lca}}) = \frac{l(S_{r_1}, n_{\text{lca}}) + l(S_{r_2}, n_{\text{lca}})}{2} \tag{3.5}$$

as the average distance from the internal node n_{lca} to the two chosen references in $\mathcal{T}_{\text{ref}}$. To determine the new edge lengths of e_{q2} and e'_q, we distinguish three situations: If $\hat{d}(S_q) < \hat{d}(n_{\text{lca}})$, we set $l(e_{q2}) = 0$, $l(e_{q1}) = l(e_q)$, and the length of e'_q is set to $l(e'_q) = 0$. In this case, no placement position with positive branch lengths adequately reflects the calculated distances. Setting a negative branch length for the pendant branch would comply with the distance estimates; however, we decided to simply reduce the branch length to zero, since negative branch lengths limit subsequent processing and lack a clear biological meaning. If $\hat{d}(n_{\text{lca}}) \leq \hat{d}(S_q) \leq 2 \cdot l(e_q) + \hat{d}(n_{\text{lca}})$, the lengths of e_{q2} and e'_q are set to

$$l(e_{q2}) = l(e'_q) = \frac{\hat{d}(S_q) - \hat{d}(n_{\text{lca}})}{2} \tag{3.6}$$

and, as before, $l(e_{q1})$ is set to $l(e_q) - l(e_{q2})$: The placement of S_q fits well onto the selected branch with respect to the estimated mean distance $\hat{d}(S_q)$ and the branch lengths in $\mathcal{T}_{\text{ref}}$. In the last case $\hat{d}(S_q) > \hat{d}(n_{\text{lca}}) + 2 \cdot l(e_q)$ is satisfied and the heuristic sets $l(e_{q2}) = l(e_q)$,

$l(e_{q1}) = 0$, and $l(e'_q) = \hat{d}(S_q) - \hat{d}(n_{\text{lca}}) - l(e_q)$. Again, an implicit trifurcation with branch lengths of 0 occurs in the first and last case.

LCA-Count — As in the previous approach, queries are placed only at inner nodes. But instead of using the reference sequences S_{r_1} and S_{r_2} that minimize the distance to S_q, we select the two references S_{r_1} and S_{r_2} with the maximum number of spaced-word matches to S_q with scores larger than t. We then calculate the lengths of the newly generated edges as in LCA-Dist.

SpaM-X — The four previous heuristics produce placements only on branches directly above the leaf nodes, or only on branches above inner nodes of $\mathcal{T}_{\text{ref}}$. This is not optimal, as placements for a set of query sequences are likely to occur across the whole reference tree. Thus, SpaM-X combines the previous approaches: For the most similar reference S_{r_1} and the second most similar reference S_{r_2} according to the distance estimates inferred from spaced-word matches, SpaM-X evaluates whether

$$|s(S_q, S_{r_1}) - s(S_q, S_{r_2})| > \frac{s(S_q, S_{r_1}) + s(S_q, S_{r_2})}{X} \tag{3.7}$$

holds. If so, S_q is placed directly above the reference S_{r_1} according to SpaM-Count. Otherwise, S_q is placed according to LCA-Count. Consequently, the SpaM-X heuristic places S_q above the leaf annotated with S_{r_1} if and only if the number of matches between S_q and S_{r_1} is substantially higher than the number of matches to the next similar sequence S_{r_2}. If the difference in the number of matches is small, the query is placed above their lowest common ancestor instead. A disadvantage of this heuristic is that it requires a hyper-parameter X. We set $X = 4$ as default and refer to the heuristic as SpaM-4. Possible choices for X and how to infer it experimentally are discussed below.

SpaM+APPLES — In addition to the five placement heuristics above that are natively included in App-SpaM, we also use the least-squares optimization criterion of APPLES to calculate placement positions. For this, we forward a matrix with our distance estimates $d(S_q, S_r)$, $r \in [m]$ for all queries to APPLES. APPLES then finds the position within $\mathcal{T}_{\text{ref}}$ such that the sum of squared differences between estimated distances and distances present in the tree is minimal, as explained in Subsec. 2.7.1. Using APPLES has the advantage that placements are inferred with a solid framework of phylogenetic tree inference. However, it comes with increased computational demands and requires that the distances of $\mathcal{T}_{\text{ref}}$ are in accordance with the distance estimates inferred by App-SpaM; this means that branch lengths of $\mathcal{T}_{\text{ref}}$ have to correspond to the average number of nucleotide substitutions under the Jukes-Cantor model of sequence evolution. If the origin of $\mathcal{T}_{\text{ref}}$ and, thus, the nature of the branch lengths is unknown, they can be re-estimated on the basis of an existing alignment using programs such as FastTree [347]. However, this again requires the existence of a reference MSA, limiting the applicability of this heuristic.

Several additional caveats apply to the established heuristics: $\mathcal{T}_{\text{ref}}$ must be *rooted* in order for LCA-Dist, LCA-Count, and SpaM-X to work. Only when $\mathcal{T}_{\text{ref}}$ is rooted, the lowest common ancestor is uniquely defined. We expect any end user to supply a rooted tree; if no root is specified for the input tree, App-SpaM automatically roots $\mathcal{T}_{\text{ref}}$ at its midpoint and warns the user that placement results should be taken with caution. Furthermore, there is the hypothetical possibility that no match is found for a query S_q. We try to prevent this from happening by choosing a short word length $l = 44$ with $w = 12$ match positions. Still, it might occur for distantly related or erroneous query sequences. In the event that no spaced-word matches are found between S_q and any reference, we place S_q at the root of $\mathcal{T}_{\text{ref}}$ with a predefined default branch length for the pendant branch. The existence of such placements for multiple query sequences could indicate that the set of references is insufficient and that it does not represent a good sample among the species present in the query sequences. Accordingly, we observed that ML-based placement programs do not find sensible placement locations for queries where App-SpaM detects zero spaced-word matches. The complete algorithmic design of App-SpaM is outlined in detail in Fig. 3.1.

App-SpaM works on DNA sequences over the alphabet $\Sigma = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. Each of the four symbols is internally encoded in 2 bits, `A` as `00`, `C` as `01`, `G` as `10`, and `T` as `11`. Thus, we can represent up to 32 DNA symbols with a single 64-bit integer value. Limiting the number of don't care positions to 32 entails that the memory requirements for spaced words and spaced-word matches as well as the overhead to maintain and handle the associated data structures stay as minimal as possible. The data structure to save a single word in App-SpaM uses a total of 28 bytes: 8 bytes each to save the match and don't care positions, respectively, and another 12 bytes to save auxiliary information about the words including in which sequence and at which position within the sequence they were found. All spaced words that contain symbols that are not present in Σ are removed from further consideration all together. Saving the match and don't care positions explicitly within the spaced-word data structure has the drawback that large amounts of information are stored repeatedly. Each two bits that encode a single symbol of an input sequence are saved in a total of l spaced words for a pattern length of l. In contrast, it is also conceivable that a spaced-word data structure only saves its match position, the sequence from which it originates, and the location within this sequence (and it is implemented in this manner by FSMW). Although this only requires 12 bytes for each spaced word, the strategy used in App-SpaM has large computational gains later on: When evaluating the don't care positions of identical words, it is not required anymore to look up the nucleotides at the original sequence positions. Such a lookup constitutes a bottleneck, especially when $\mathcal{T}_{\text{ref}}$ consists of many references, as entirely different sequence regions must be stored in main memory and are accessed consecutively. The computer system cannot perform any cache optimization for such unstructured accesses, as it is unknown from which sequences consecutive spaced words originate. By saving the don't care positions together with each spaced in consecutive word blocks, no such lookup has to take place. When the lists of identical spaced words from query and reference sequences are traversed simultaneously, the don't care positions are retrieved immediately and their score is calculated straight away. Still, our method runs into computational difficulties when large amounts of long reference sequences are present. To counteract this, we use a sampling strategy proposed in Sec. 4.3.

The list L_r that contains spaced words from all references is computed once and stored in main memory because it is accessed for each query sequence. In contrast, the spaced words for a query are required only once, and there are potentially large amounts of query sequences in metataxonomic use cases. Thus, to limit the memory consumption, the query sequences are divided into *batches* and each batch is processed consecutively. When a query batch is processed, the according list L_q of all spaced words from this batch is created and placement positions are inferred as described earlier. Then, the list L_q is deleted and the next batch is processed. This also allows for a straightforward parallelization strategy: Multiple query batches are processed simultaneously by multiple computing cores without interfering with each other since queries do not influence each others placement position. App-SpaM uses a default batch size of 100 000 query sequences.

We evaluated the accuracy and speed of App-SpaM compared to other state-of-the-art programs using PEWO. PEWO is a framework that provides reproducible evaluation workflows for phylogenetic placement that allow for a standardized evaluation process across multiple data sets and programs. Internally, it relies on the workflow tool Snakemake [348] and uses Conda to manage the evaluation environment. The installation of PEWO also comes with the most recent software version of all associated placement tools; currently, these tools include pplacer, EPA, EPA-ng, RAPPAS, APPLES, and App-SpaM. The PEWO repository is mainly intended to host the evaluation workflows and is not designated as a data repository; still, it includes several data sets that are ready to be used for evaluation purposes. PEWO is a recent development and has not been used for any comprehensive evaluation of multiple or all placement programs so far. Thus, the conducted evaluation not only serves as a means of evaluating App-SpaM, but is also a first step toward a common benchmark including all currently available programs.

The pruning-based accuracy evaluation (PAC) workflow provided by PEWO assesses placement accuracy, while the resources evaluation (RES) workflow identifies the time and memory requirements of PP software programs. Additionally, there exists a likelihood-based accuracy evaluation (LAC) workflow that is intended for ML-based methods. As described in Subsec. 2.7.3, the PAC workflow follows a pruning-based process: The user specifies a data set consisting of the reference tree $\mathcal{T}_{\text{ref}}$ and the corresponding MSA $\mathcal{A}_{\text{ref}}$, the programs to be evaluated and their parameters to be tested, and a fixed number of iterations. In each iteration, PEWO selects a random node of $\mathcal{T}_{\text{ref}}$ and prunes the entire subtree attached at this node. The corresponding reference sequences of the pruned leaves are removed from $\mathcal{A}_{\text{ref}}$ as well. The branch to which the pruned subtree was attached is the *expected* placement branch and represents the optimal placement location for every pruned sequence. Then, query reads of a specified length are simulated from all pruned sequences by simply splitting the sequences into shorter non-overlapping segments. By default, reads with a length of 150 bp are created, and resulting reads with a length below 50 bp are dropped. The query reads are placed back onto the pruned tree by each specified program with all chosen parameter combinations. We call the placement branch determined by a program its *proposed* placement branch. Some PP programs output multiple proposed placement branches for each single query weighted by the according likelihood values. The proposed placements of each query are compared to the expected placement location using two metrics: The node distance (ND) measures the number of nodes between the expected placement branch and the proposed placement branch; only the branch with the highest weight is considered when multiple placement branches are specified. The expected node distance (eND) calculates the average number of nodes across all proposed placement branches for a query weighted by their likelihood-weight-ratios.

We used ten data sets for the evaluations; an overview of them is provided in Tab. 3.1. These data sets were compiled from different sources and vary with respect to their size (number of reference sequences and their length), locus, tree structure, and composition of reference sequences. We refer to each data set by an abbreviation consisting of its name followed by the number of reference sequences it comprises. The main focus of this evaluation is to compare App-SpaM to existing tools for PP with respect to its accuracy and computational demands; as several of the other tools rely on alignments and involve computationally expensive ML-calculations, the selection of data sets is oriented towards the exemplary use case of short read assignment by means of phylogenetic placement. Thus, the sequences span single marker genes which are handled effortlessly by all PP programs. We also apply App-SpaM to other use cases of PP that involve more diverse data sets, see below and Chpt. 5.

The first two data sets (bac-150, hiv-104) were acquired from the PEWO repository. bac-150 represents a typical metataxonomic setting in which 16S sequences are used to identify query sequences. The latter (hiv-104) is a collection of complete HIV genomes. Three additional data sets (neotrop-512 [349], tara-3748 [350], and bv-797 [351]) were taken

Table 3.1 – Data sets used in the PAC and RES evaluations.

abbreviation	locus	mean length (bp)	query length (bp)
BAC-150	16S	1 256	150
HIV-104	viral genomes	9 096	150, 500
NEOTROP-512	16S	1 766	150, 300
TARA-3748	16S	1 406	150, 300
BV-797	16S	1 341	150
EPA-218	16S	1 483	150
EPA-628	5.8S	780	150
EPA-714	16S	1 169	150
WOL-43	microbial genomes	52 768 066	150
CPU-652	16S	1 315	150
CPU-512	16S	1 766	150

Each data set is abbreviated by its name and the number of reference sequences that it comprises. In their order of appearance, the columns present the abbreviation of the data sets, the locus from which the sequences originate, the mean sequence length of the references, and the simulated read lengths chosen for the evaluation. The first eight data sets are used in the *PAC* workflow, WOL-43 is used to evaluate APP-SPAM, and the *RES* workflow is applied to the last two data sets. This table is adapted from *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

from the accuracy evaluation carried out by EPA-NG; they also consist of 16S sequences. The next three data sets (EPA-218, EPA-628, EPA-714) comprise the marker genes 16S or 5.8S and originate from the EPA evaluation. The WOL-43 data set consists of 43 complete genomes of different Wolbachia species and was only used for APP-SPAM due to its size (the average reference sequence length is 52 768 066 bp). The last two data sets (CPU-652 and CPU-512) assess computational demands; the first one is also taken from the PEWO repository, while the second one is identical to the NEOTROP-512 data set.

We performed several test runs with the PAC and RES workflows on the data sets to examine the capabilities and limits of APP-SPAM with respect to placement accuracy, speed, and memory requirements. First, we evaluated APP-SPAM under varying settings to analyze the influence of the choice of parameters on its placement accuracy. For this, we ran APP-SPAM on the BAC-150 and HIV-104 data sets with varying pattern weights $w \in \{8, 12, 16\}$ and all five placement heuristics. For SPAM-X we used $X = 2$ and $X = 4$. We also ran SPAM+APPLES on the BAC-150 data set. The number of don't care positions was always fixed at 32. The length of the simulated query reads was set to 150 bp for the BAC-150 data set comprising short 16S sequences and to 150 bp or 500 bp for the longer viral genomes of the HIV-104 data set. For each combination of parameters, 100 random prunings were performed with PEWO. In each pruning, a random node within $\mathcal{T}_{\text{ref}}$ is chosen and the subtree below it is pruned. Thus, prunings vary with respect to the number of pruned reference sequences and the size of the remaining tree. Most prunings comprise only a single or very few reference sequences (half of the nodes of a balanced bifurcating tree are its leaves), but some prunings also comprise more than half of the original sequences. This represents any real-world example reasonably well when it is assumed that the chosen reference tree comprises a good

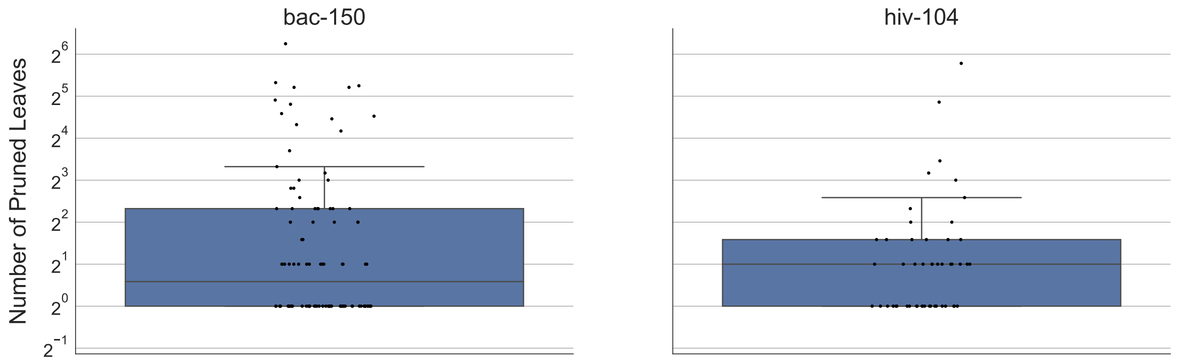


Figure 3.2 – Number of pruned leaves (y-axis, log-scale) for the BAC-150 and HIV-104 data sets (x-axis) and 100 pruning events. The number of removed leaves is illustrated for each pruning event (black dots) and statistics across all 100 prunings are indicated (box plots). Since half of the nodes in any balanced tree are its leaves, most pruning events contain only a single reference sequence.

representation of the species that are also present in the queries. Under this assumption and a largely balanced reference tree, most query sequences have closely related references (instances in the simulation where single sequences are pruned), while cases with no closely related references are rare (large pruning events in the simulation that comprise many references). Figure 3.2 displays the number of pruned leaves for each of the 100 pruning events for the data sets BAC-150 and HIV-104 as an example. The random prunings for all other data sets are expected to exhibit a comparable distribution.

App-SpaM is able to use multiple spaced-word patterns simultaneously: If this option is turned on, it invokes RASBHARI to create the specified number of patterns while minimizing their overlap complexity. RASBHARI is initialized with a random seed and, thus, different runs of App-SpaM will use different pattern sets. We tested the influence of the number of patterns on the placement accuracy as well as the variance across different pattern sets of identical size. For this, we ran App-SpaM with the LCA-Count and SpaM-4 heuristics on the BAC-150 and BV-797 data sets and varied the size of the pattern set from a single pattern up to five patterns with 100 random pruning events each. We repeated each experiment with five different pattern sets using different random seeds to initialize RASBHARI to assess the variance with respect to the structure of the patterns.

In addition to running App-SpaM on its own, we conducted a comprehensive evaluation of the placement accuracy of all placement programs currently available in PEWO (PPLACER, EPA, EPA-NG, RAPPAS, APPLES, and App-SpaM) in the metataxonomic setting. For this, we ran the PAC workflow for each data set (except WOL-43 due to its size, and the two data sets intended for the RES workflow) including each program with a variety of program parameters. Seven of the eight data sets comprise a single marker gene and thus have a similar sequence alignment length. The eighth data set comprises the 104 viral genomes and its sequences are longer by approximately a factor of eight. However, the data sets vary considerably with respect to the group of organisms they comprise, the number of reference sequences that are available, and the degree of similarity between reference sequences, see Suppl. Tab. A.1. For each program, the parameters were chosen according to their manuals and additional instructions provided by PEWO. A complete table of all parameter combinations is to be found in Suppl. Tab. A.2. The read length was fixed at 150 bp, a common length produced by popular Illumina sequencing platforms [352]; additionally we performed experiments with a query read length of 500 bp for HIV-104, and a length of 300 bp for NEOTROP-512 and TARA-3748. Again, 100 pruning events were conducted for each experiment, respectively.

In all experiments so far, the query reads are created by simply splitting the pruned references into reads of a predefined fixed length. As a result, the queries do not exhibit errors or biases that would normally occur from the sequencing preparation, the sequencing itself, or common preprocessing steps. We extended PEWO with the functionality to simulate query sequences from the pruned references using error profiles of common sequencing technologies [353]. For this, we integrated ART [341] into the existing PAC workflow so that it simulates a fixed number of queries for each pruned reference sequence. With this setup, we ran all PP programs with default parameters on the BAC-150, HIV-104, and NEOTROP-512 data sets with 50 pruning events, respectively. We used the Illumina *HiSeq 2500* error profile of ART with its default parameters to simulate 50 query reads for each pruned reference sequence. For most data sets, this corresponds to a coverage of approximately five; we assume that increasing the number of query sequences beyond that would not alter the observed placement accuracy.

Furthermore, we tested APP-SPAM's unique capability of working with unassembled reference sequences. With the general increase in sequencing efforts comes an ever-growing amount of sequence data that remain without assembly [354]; this includes scaffold bins from metagenome assembled genomes [9, 355], short reads thereof, or NGS single cell DNA or RNA sequencing data [356]. We evaluated the performance of APP-SPAM with regard to unassembled reference sequences with varying coverages on HIV-104 and WOL-43. We transformed both data sets so that each reference consists of a bin of short query reads with a length of 150 bp with a coverage $C \in \{4, 2, 1, 0.5, 0.25, 0.125, 0.0625, 0.03125\}$. Reads were simulated by repeatedly drawing 150 consecutive nucleotides, starting at random positions of the original sequence until the specified coverage was reached. As PEWO does not support this kind of data, we implemented a simple leave-one-out pipeline to assess the placement accuracy: In every pruning event, a bin of a single reference sequence is removed from the data set and the according reference is pruned from the tree. Then, the according reads are placed back onto the tree and the average ND is measured across all reads. The procedure is repeated for every reference sequence in the data set.

We used PEWO's RES workflow to measure the runtime (preprocessing and main computations) and peak memory usage of all six programs on the two data sets CPU-652 and CPU-512. The preprocessing steps do not include the creation of the reference alignment $\mathcal{A}_{\text{ref}}$, but only the alignment of query sequences against $\mathcal{A}_{\text{ref}}$ or comparable steps such as the creation of the phylo-k-mer database. Instead of a pruning-based evaluation, PEWO simply places a set of supplied query reads onto the reference phylogeny. In five repetitions, we placed 100 000 query reads on the CPU-652 data set and 10 000 query sequences on the CPU-512 data set. The number of query sequences in these experiments is chosen rather low due to excessive runtimes of certain ML-based programs. Thus, we also run APP-SPAM on the largest metataxonomic data set TARA-3748 by simulating $3748 \cdot 10^n$, $n \in \{1, 2, 3, 4\}$ reads of length 150 bp for each reference, resulting in up to 37 480 000 query reads. We placed the reads with APP-SPAM using weights of $w = 12$ and $w = 16$ using 30 cores concurrently. Each experiment was performed twice. All of the above experiments were carried out on *Intel(R) Xeon(R) E7-4850* CPUs with 2 GHz.

3.1.2 Evaluation Results

Unlike programs based on ML, App-SpaM and APPLES do not specify multiple placement branches for a single query S_q. Such weighted placement positions provide information about the placement uncertainty of S_q. This is helpful in certain instances, for example, when high accuracy is desired and uncertain placements shall be discarded from subsequent analysis. However, the first placement position with the highest weight is sufficient in many cases: for example, to uniquely annotate queries or when visualizing placements of multiple query sequences. Thus, we primarily report node distance (ND) values throughout this section, as they allow a direct comparison between all involved programs. In most cases, the expected ND (eND) and ND values for any program hardly differ; ML-based programs often perform slightly better under the eND metric than under the ND metric. We point out those instances where eND values deviate substantially from the provided ND values. We always report the node distance rounded to two decimal places; the appendix contains detailed tables with exact values for all presented results, including the eND.

For each experimental setup, the placement results are presented as a combination of box plots and strip plots. Each black dot in a strip plot indicates the average ND *across all query sequences of a single pruning event*. The horizontal spread of black dots is a purely visual aid. Each box plot summarizes statistics for an experimental setup *across all pruning events*: The box itself delineates the lower and upper quartiles of the distribution, while the line in the middle indicates the mean. The extent of the remaining node distances that are contained within 1.5 times the interquartile range is indicated by the whiskers. All other data points outside the whiskers are specified as outliers.

The average performance for different heuristics with the default weight $w = 12$ across 100 pruning events on the bac-150 data set is illustrated in Fig. 3.3. SpaM-4 has the lowest average ND of 4.65, closely followed by LCA-Count with a ND of 4.73. Thus, the proposed placement position of a query is, on average, 4.65 nodes (or 4.73 nodes) away from

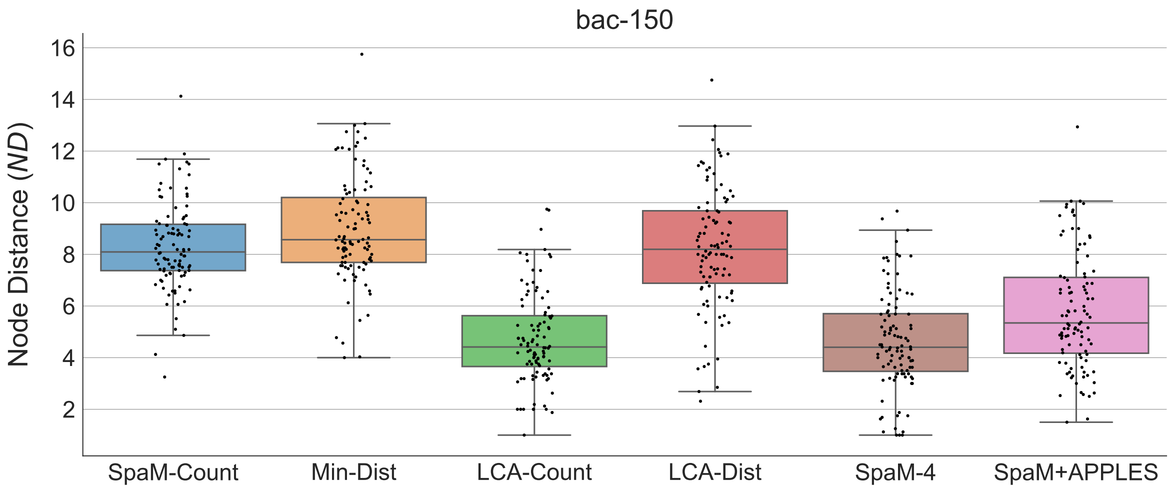


Figure 3.3 – Node distance (ND) of placements (y-axis) inferred with App-SpaM over 100 repetitions for the default pattern weight of $w = 12$ with different placement heuristics (x-axis, different colors) on the bac-150 data set. The mean ND across all queries for a single repetition is shown (black dots), as well as the distribution of NDs across all repetitions (box plots). This figure was adapted from *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

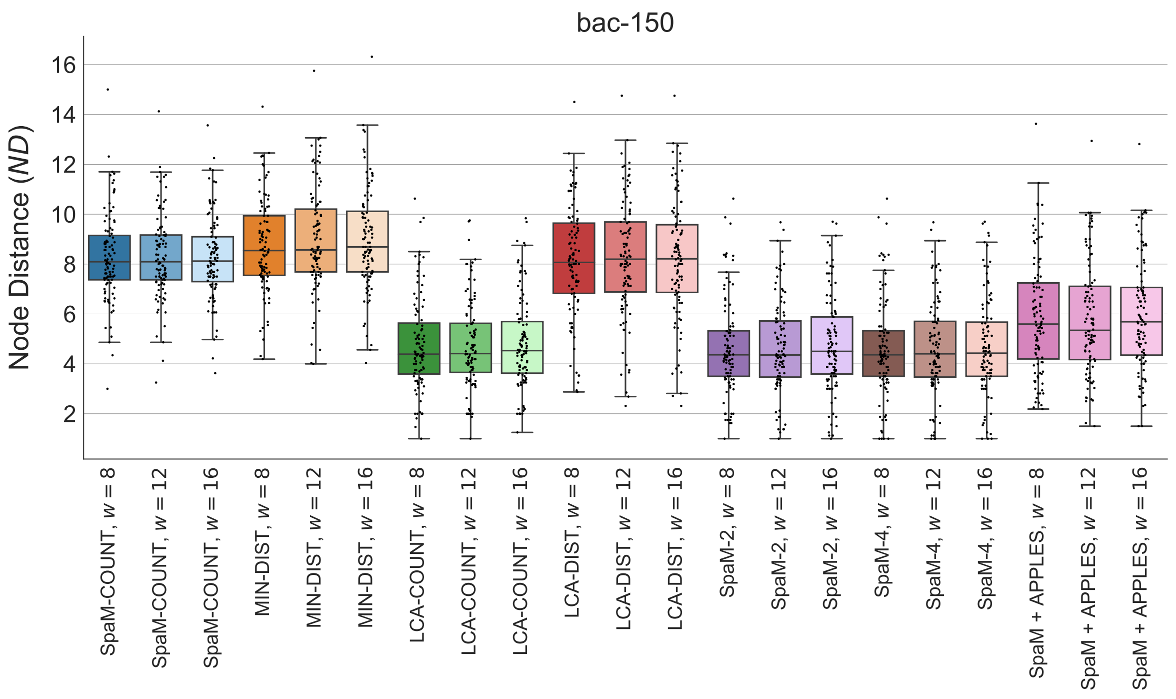
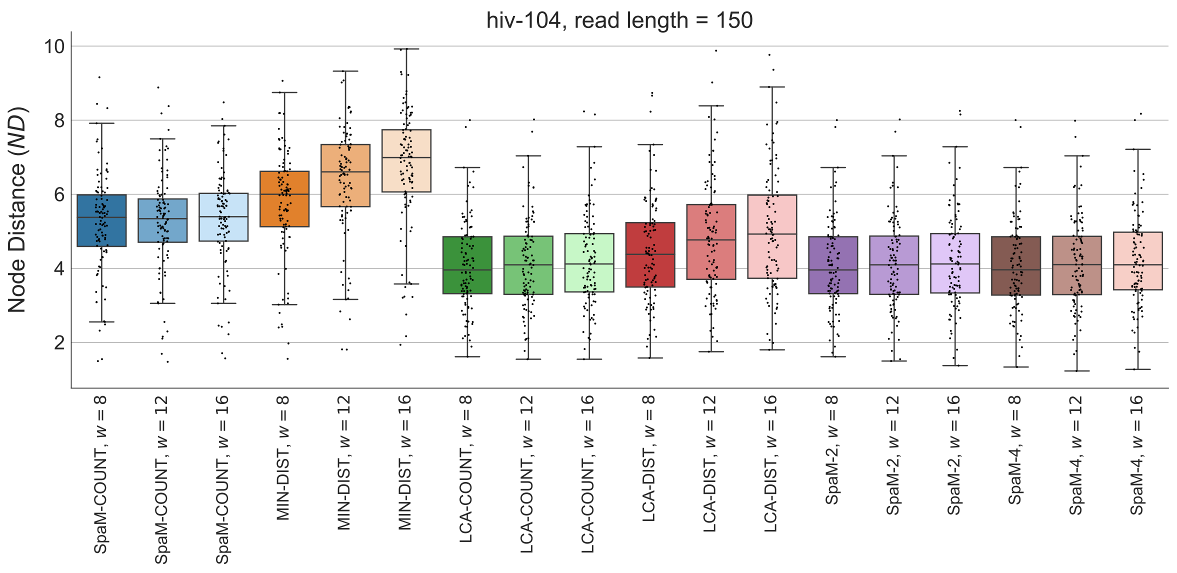


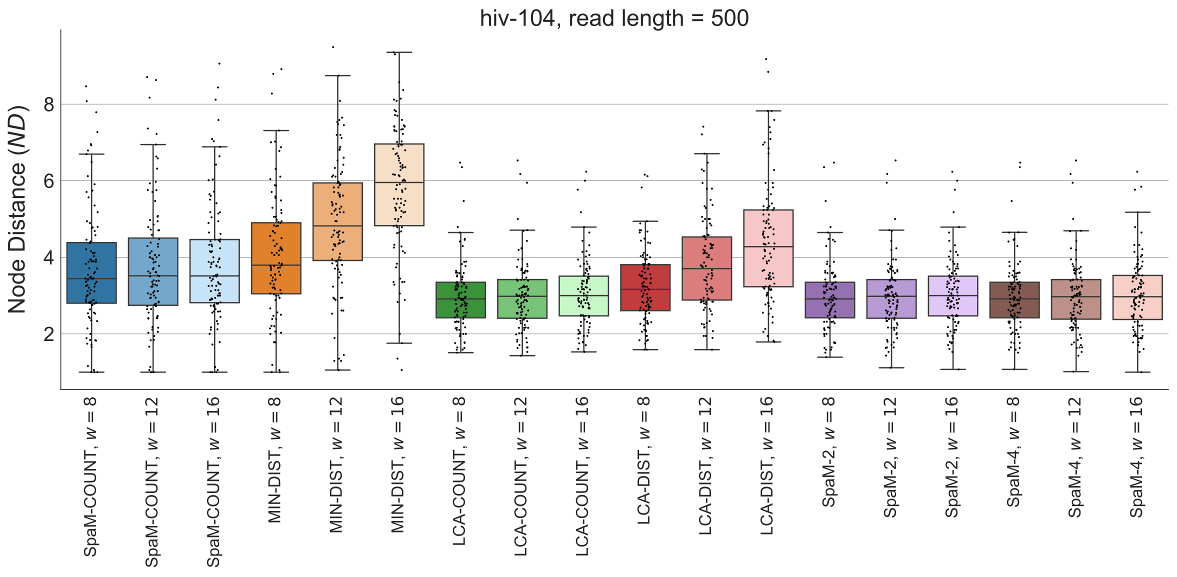
Figure 3.4 – Node distance (ND) of placements (y-axis) inferred with App-SpaM over 100 repetitions for different pattern weights $w \in \{8, 12, 16\}$ (different hues) and different placement heuristics (x-axis, different colors) on the bac-150 data set. The mean ND across all queries for a single repetition is shown (black dots), as well as the distribution of NDs across all repetitions (box plots). This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

the expected placement position in the pruned reference tree. With a mean ND of 5.81, the proposed placement positions of SpaM+APPLES are on average more than one node further away from the expected position than those placements from SpaM-4 and LCA-Count. The other three heuristics (SpaM-Count ND of 8.31 , Min-Dist ND of 8.98, LCA-Dist ND of 8.22) perform significantly worse. The exact statistics of the box plots in Fig. 3.3 are given in Suppl. Tab. A.4.

Figure 3.4 presents the accuracy results for App-SpaM on the bac-150 data set when using different pattern weights for different placement heuristics. Here, the SpaM-X heuristic is additionally displayed for $X = 2$. Although the average accuracy differs substantially among placement heuristics, it differs little between varying pattern weights, regardless of the underlying heuristic. The divergence between the best and worst performing weights is only 0.12 nodes on average. No pattern weight consistently performs the best or worst across the heuristics; $w = 8$ is the best in two instances, $w = 12$ works best in four heuristics, and $w = 16$ for one heuristic. The variance of the node distance is large in each setting: the standard deviation of the ND in each experiment is on average 2.02. Thus, the quality of placements varies considerably between different pruning events. When taking into account all combinations of pattern weights and heuristics, SpaM-4 with $w = 12$ performs again best with an average node distance of 4.65. The SpaM-4 heuristic with $w = 12$ is closely followed by all other experiments that employ SpaM-X or LCA-Count. The exact statistics for Fig. 3.4 are found in Suppl. Tab. A.3.



(a) Accuracy of different App-SpaM heuristics and varying pattern weights for a query read length of 150 bp.



(b) Accuracy of different App-SpaM heuristics and varying pattern weights for a query read length of 500 bp.

Figure 3.5 – Node distance (ND) of placements (y-axis) inferred with App-SpaM over 100 repetitions for different pattern weights w and different placement heuristics (x-axis). The mean ND across all queries for a single repetition is shown (black dots), as well as the distribution of NDs across all repetitions (box plots) on the hiv-104 data set. The upper plot (a) shows the results for a query read length of 150 bp and the lower plot (b) for an increased query read length of 500 bp. This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

To ensure that these observed results are not specific to the chosen data set and used query read lengths, we repeated the tests for hiv-104: Fig. 3.5 shows the accuracy of different

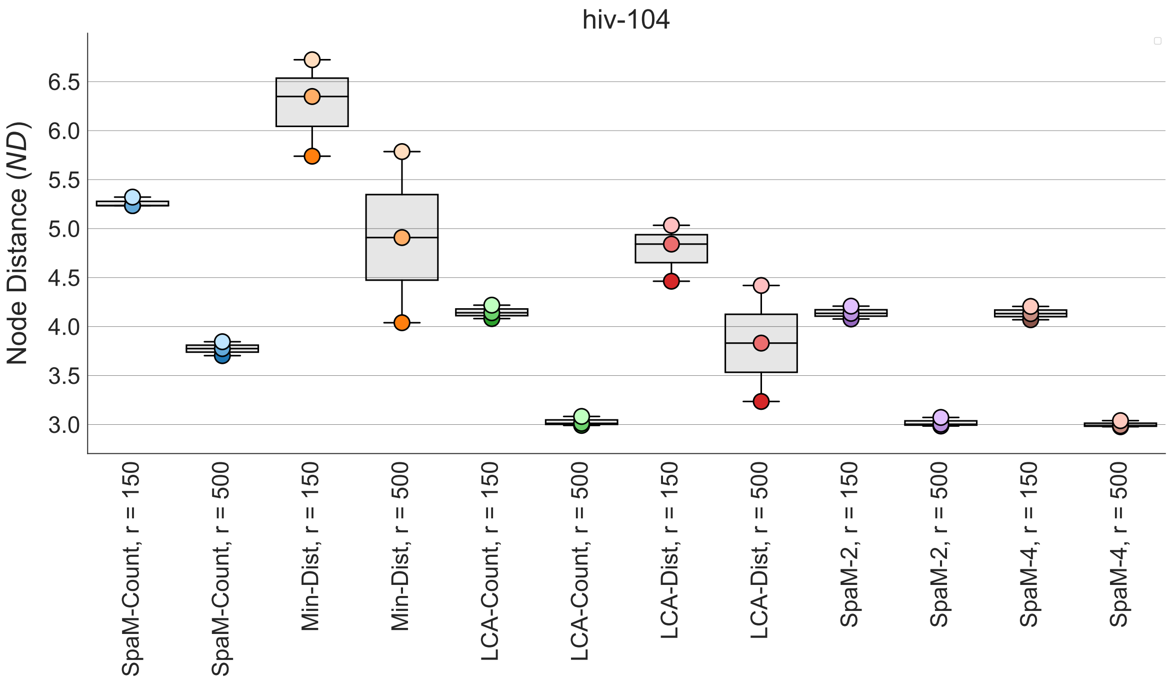


Figure 3.6 – Summary of Fig. 3.5. Average node distance (ND) of placements (y-axis) across pruning events for different pattern weights $w \in \{8, 12, 16\}$ (different hues) and different placement heuristics (x-axis, different colors) on the HIV-104 data set with query read lengths of 150 bp or 500 bp. The mean ND across all repetitions for each weight is specified (colored dots, color marking as in Fig. 3.5), as well as the distribution of average NDs across all weights (box plots).

APP-SPAM heuristics and different pattern weights across 100 random pruning events for the HIV-104 data set for a query read length of 150 bp (Fig. 3.5a) or of 500 bp (Fig. 3.5b). Here, we dropped SPAM+APPLES and only included those heuristics that are natively included in APP-SPAM. As before, SPAM-2, SPAM-4, and LCA-COUNT perform equally well and are robust for different values of the pattern weight. In contrast to the BAC-150 data set, the accuracy varies considerably for different values of w when using the MIN-DIST or LCA-DIST heuristics. For both, the average node distances increase with larger pattern weights. These results hold true irrespective of the query read length. However, all heuristics consistently perform better when query read lengths are longer, no matter which pattern weight is used.

A summary of Fig. 3.5 is given in Fig. 3.6: Here, the mean ND for all repetitions in each experiment is depicted as a single colored dot. By this, the difference of the mean accuracy and its variance with respect to the read length becomes apparent. The best parameter configuration (SPAM-4 with $w = 8$) improves from a ND of 4.07 to a node distance of 2.98 when the lengths of query reads are prolonged from 150 bp to 500 bp; a drop of approximately 27%. The mean performance also improves substantially for longer query sequences when using other heuristics. The observed effect of worsening placement locations for increasing pattern weights in the MIN-DIST and LCA-DIST heuristics is more severe for long queries. On the contrary, the weight of the pattern has little influence when used with the LCA-COUNT and SPAM-X heuristics. Based on these experiments, we assume that the SPAM-4 heuristic with $w = 12$ is a robust choice for a wide range of data sets and query lengths. Subsequent tests confirm that these parameters produce consistent results. In Fig. 3.6, $w = 8$ sometimes

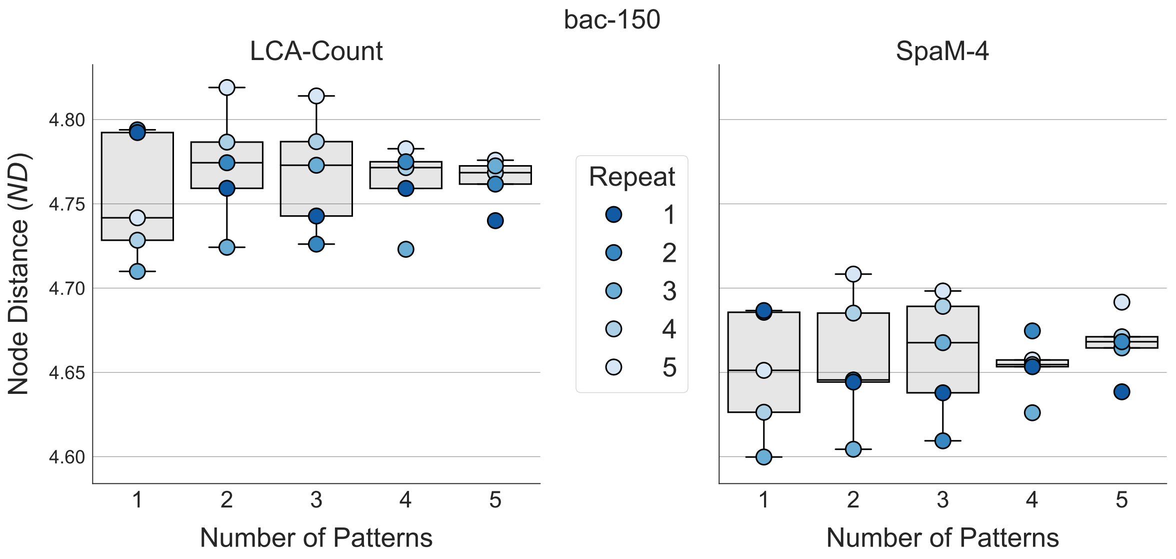


Figure 3.7 – Average accuracy measured as ND (y-axes) for the LCA-Count (left) and SpaM-4 (right) heuristics dependent on the number of patterns (x-axes). The mean of 100 iterations is given for five repetitions (blue dots, different hues) for each pattern set size. Pattern sets were compiled with different random seeds using rasbhari.

produces better results than $w = 12$; however, choosing a lower weight comes with the downside that runtimes may be significantly longer. This is due to the increase in the number of spaced-word matches that occur purely by chance rather than from homologous sequence positions. Each of these spaced words is subjected to the time-intensive filtering procedure by which it is (presumably) discarded. This effect is especially severe when whole genomes are used as references. As depicted in Fig. 3.5, the weight of the pattern has little influence on the placement accuracy when used together with the LCA-Count or SpaM-X heuristics. However, it is important to also analyze the influence of the size of the pattern set and the pattern structures themselves on the placement accuracy. Here, the internal structure of a pattern P refers to the exact locations of the match positions within P. For the default parameters $w = 12$ and $l = 44$ there are a total of 21 090 682 613 different patterns from which rasbhari chooses a subset of the specified size.

Figure 3.7 illustrates the variation in placement accuracy when pattern sets of varying sizes are used. The evaluation was carried out with App-SpaM on the bac-150 data set using the heuristics LCA-Count and SpaM-4 with $w = 12$ and $l = 44$. Here, a *repeat* refers to a single experiment with a fixed random seed that comprises 100 pruning *repetitions*. The figure illustrates the average ND of each repeat across all repetitions and the box plots indicate the variation between repetitions with five sets of identical pattern sizes generated from different random seeds. Additionally, Fig. 3.8 shows the overall statistics for all 100 repetitions for each repeat in more detail. The accuracy of App-SpaM barely changes across all repeats. The standard deviation of the mean node distance across all repeats is merely 0.06. Thus, neither the specific patterns used nor the number of patterns has a notable influence on the placement accuracy for the bac-150 data set. The described trends hold also true for the bv-797 data set, see Suppl. Fig. A.1 and the statistics in Suppl. Tab. A.7 and Suppl. Tab. A.8.

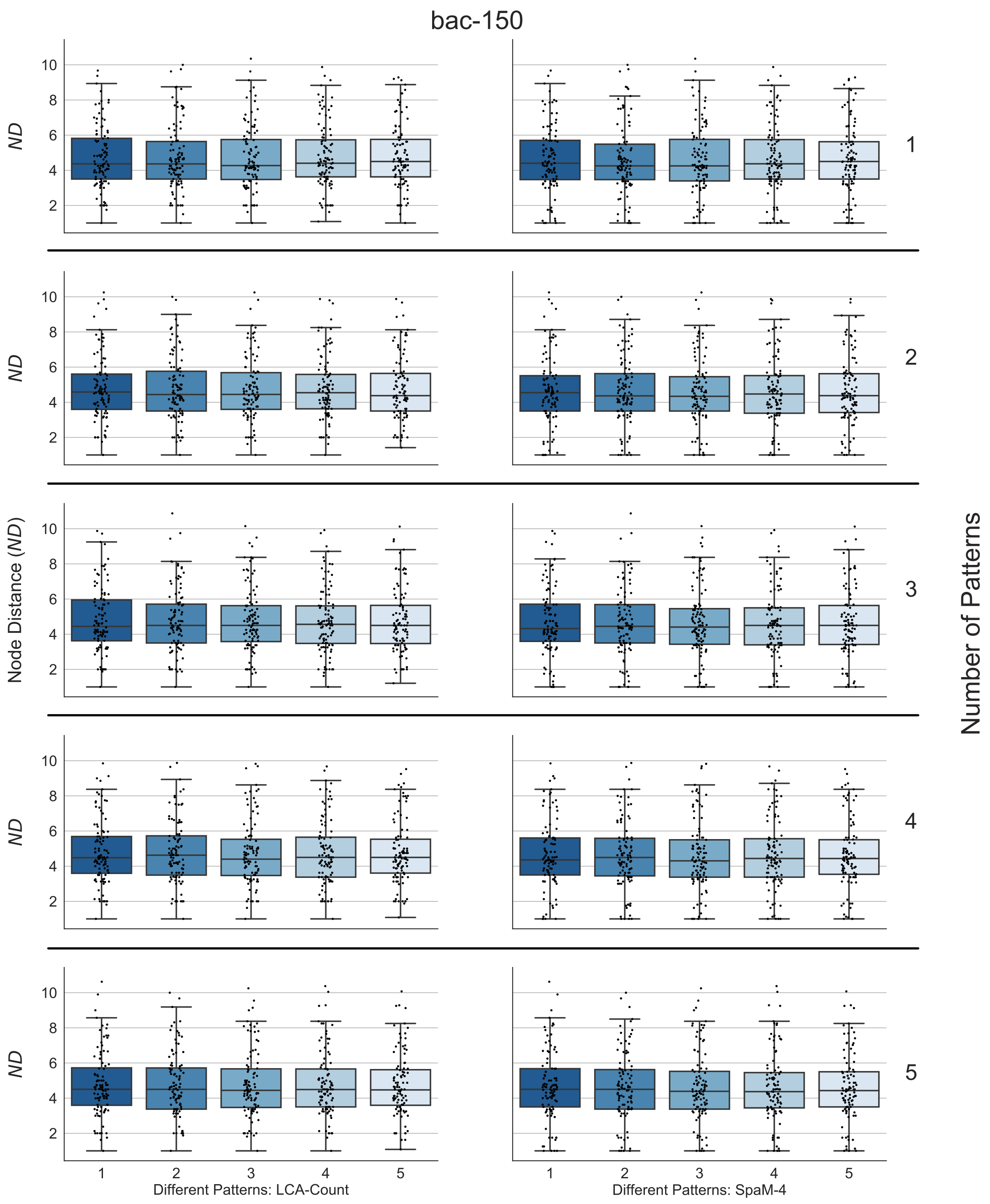


Figure 3.8 – Accuracy measured as ND (left y-axes) of the LCA-Count and SpaM-4 heuristics (outer x-axis) dependent on the number of used patterns (right y-axis). Five repetitions for each pattern set size are shown (inner x-axes, different hues). This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

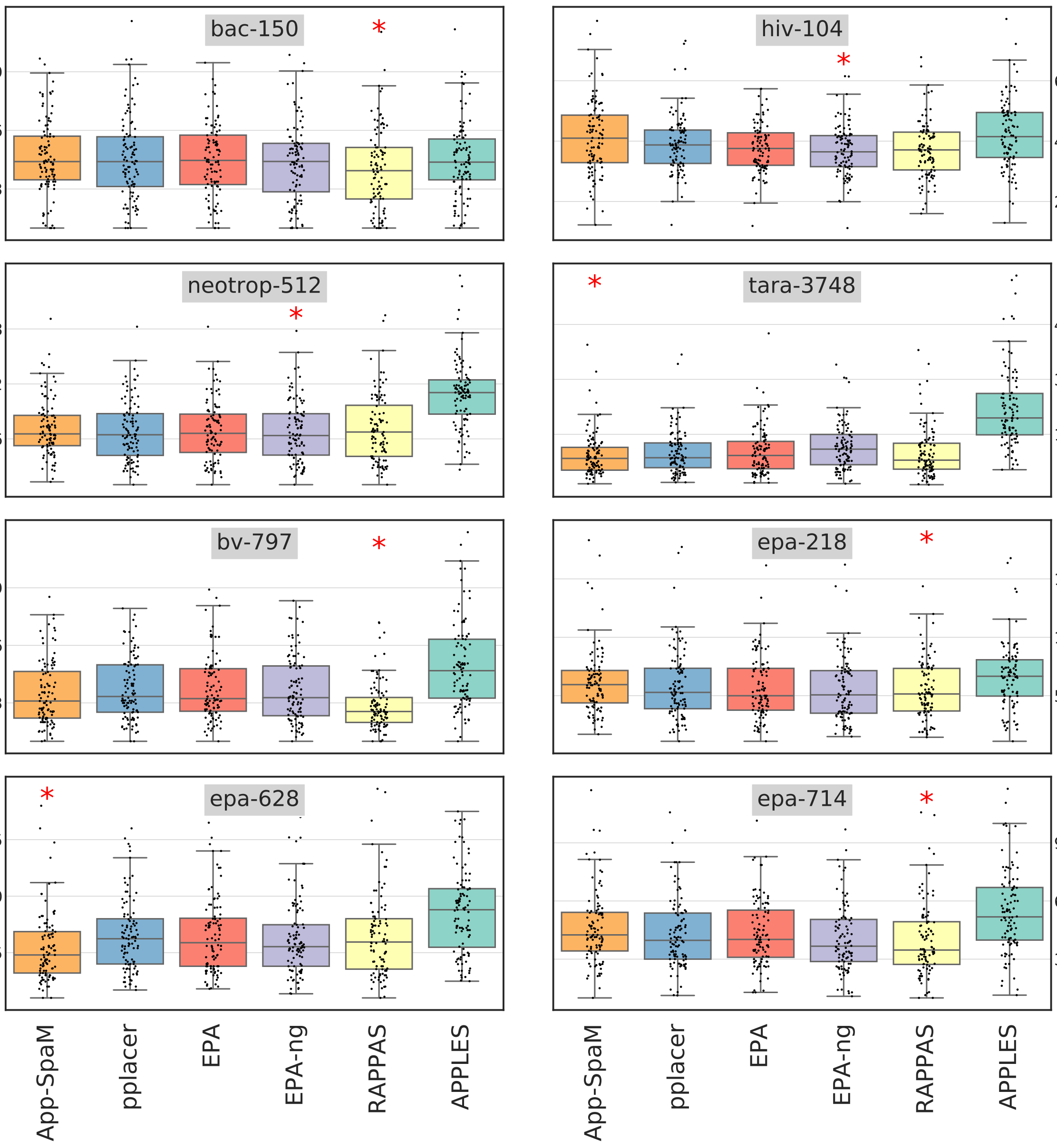


Figure 3.9 – Accuracy measured as ND for six PP programs (x-axes) on eight data sets (individual boxes). The ND scales (y-axes) differ between the data sets due to the large variation in the size of the reference trees and, accordingly, the resulting node distances. The average accuracy across all queries for each pruning event is shown (black dots) as well as overall statistics for each program on each data set across all 100 pruning events (colored box plots). The best performing program (lowest mean ND) for each data set is highlighted (red star). This figure was adapted from *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

A summary of the comprehensive accuracy evaluation, including all six placement programs, is provided in Fig. 3.9. The figure displays the accuracy for all eight data sets with the default parameters of each program as indicated in Tab. 3.2. The absolute accuracy of all programs varies strongly between data sets due to the differing number of reference sequences. Generally, large reference trees entail a higher ND for all programs. RAPPAS performs best on average

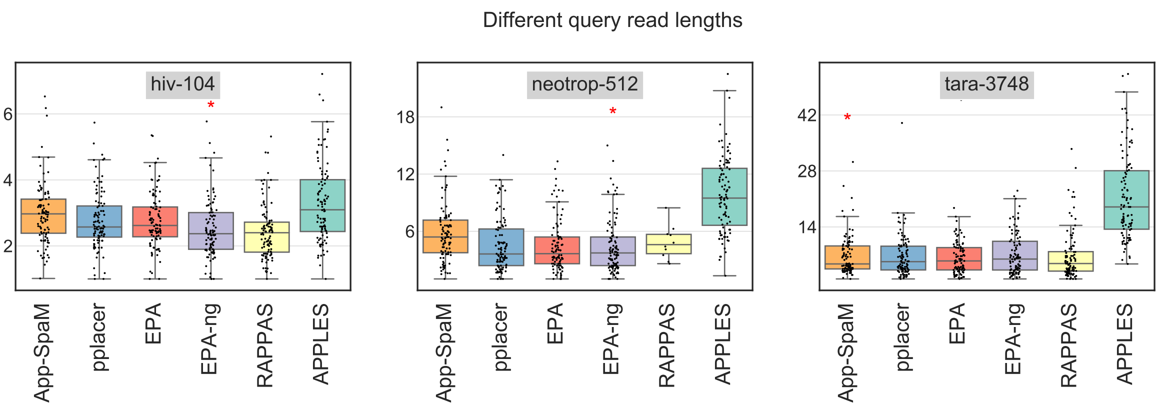


Figure 3.10 – Accuracy measured as the ND (y-axes) for six PP programs (x-axes) on three data sets (individual boxes) with query read length other than 150 bp. The read lengths are set to 500 bp for HIV-104, 300 bp for NEOTROP-512, and 300 bp for TARA-3748). This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

and achieves the lowest average ND on the BAC-150, BV-797, EPA-218, and EPA-714 data sets. EPA-NG has the lowest average ND on the NEOTROP-512 and HIV-104 data sets. On the EPA-628 and TARA-3748 data sets, APP-SPAM performs best. The mean accuracy of APP-SPAM across all data sets is 0.04 nodes better than the mean accuracy of all placement programs across all data sets; if APPLES is left out from this consideration, APP-SPAM is on average 0.01 nodes worse than the mean accuracy of all other placement programs. Thus, on average APP-SPAM performs slightly better (or slightly worse when APPLES is neglected) than the other available programs. For each data set, the programs demonstrate a similar standard deviation across the 100 pruning events. One exception is the HIV-104 data set; here, APP-SPAM's mean accuracy is sub-par and its standard deviation across the pruning events is large.

Fig. 3.10 shows the accuracy for read lengths other than 150 bp on three data sets. Due to excessive memory consumption, only 5 repetitions were performed for RAPPAS on the NEOTROP-512 data set. The results closely follow those observations of Fig. 3.9. EPA-NG performs best on the HIV-104 and NEOTROP-512 data sets, and APP-SPAM performs best on the TARA-3748 data set. In general, all programs perform better when the length of the query

Table 3.2 – Default program parameters.

APP-SPAM	RAPPAS	APPLES	EPA-NG	EPA	PPLACER
$w = 12$	$k = 8$	method: FM	heuristic: 1	$g = 0.1$	$ms = 6$
	omega: 1.5				$sb = 3$
m: EXP-4	reduction: 0.99	criteria: MLSE	$g = 0.999$		$mp = 40$

Default parameters that were used to generate the results displayed in Fig. 3.9 and Fig. 3.10. For additional parameter combinations for each program and results thereof, we refer to the appendix. This table was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

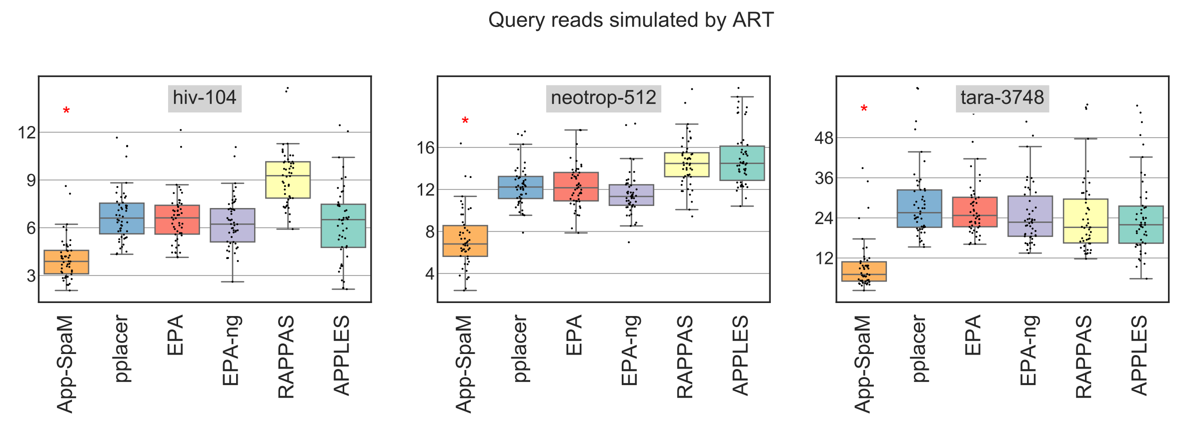


Figure 3.11 – Accuracy of PP programs when using query reads simulated with ART. The accuracy is measured as the ND for six PP programs (x-axes) on three data sets (individual boxes). The ND scales (y-axes) differ between the data sets due to the large variation in the size of the reference trees, and accordingly the resulting node distances. The average accuracy across all queries for each pruning event is shown (black dots) as well as overall statistics for each program on each data set (colored box plots) across all 50 pruning events. The best performing program (lowest mean ND) for each data set is highlighted (red star). This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

reads increases: More data is available to infer superior placement positions. For App-SpaM specifically, longer query sequences entail significantly more spaced words that can produce homolog matches to the reference sequences. The decrease in ND for longer reads is strongest for RAPPAS (average ND drops 36% across the three data sets) and less pronounced for App-SpaM (25%) and APPLES (11%). The other ML-based programs (pplacer, EPA, EPA-ng) also have a greater benefit from longer queries than the alignment-free approaches.

The results obtained for query reads simulated with ART are presented in Fig. 3.11. Unlike the default version of PEWO, ART simulates a specified number of query reads *with sequencing errors* from the set of pruned reference sequences. We simulated 50 query reads of each pruned sequence for 50 pruning events. The results deviate severely from those of the PEWO framework illustrated in Fig. 3.9: The accuracy of App-SpaM is almost not affected by the introduced sequencing errors and the mean accuracy closely follows the mean accuracy from before. The loss of accuracy amounts to 0.18% averaged over all three data sets. However, all other PP programs exhibit a significant drop in placement accuracy when query reads are simulated by ART. The average ND roughly doubles for the other programs compared to using reads from the PEWO workflow. For RAPPAS, the ND even increases by 146% from an average of 3.73 to 9.18 nodes on the hiv-104 data set. These results should be taken with caution, as discussed in more detail in Sec. 3.1.3.

We performed phylogenetic placement with App-SpaM on sets of unassembled reference sequences of varying coverages as a proof of concept. For this, we used the two data sets hiv-104 and wol-43 consisting of long reference sequences (9 096 bp and 52 768 066 bp on average, respectively) and sampled reads of length 150 bp for a range of coverages. In each experiment, each reference was then represented by a bag of reads of a given coverage. App-SpaM was run with default parameters for all experiments. Using low coverages drastically reduces the number of reads for each reference sequence, which results in fewer spaced words. Thus, we

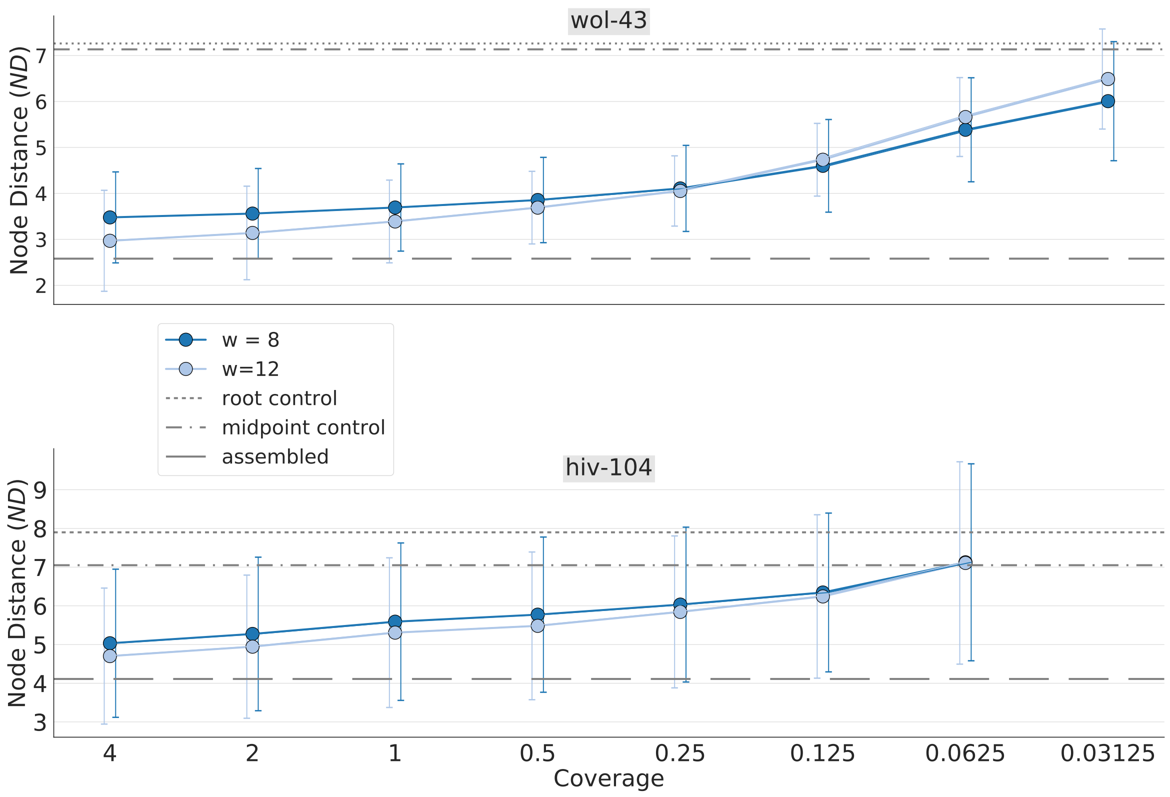


Figure 3.12 – Accuracy of App-SpaM on unassembled reference sequences. The accuracy is measured as the mean ND (y-axes) with respect to the reference coverage (x-axes) on two data sets (wol-43 and hiv-104). Error bars indicate standard deviations across 50 repetitions for each experiment. Three controls are supplied: The accuracy when App-SpaM is run on assembled references (assembled), the accuracy when each query is placed at the root of $\mathcal{T}_{\text{ref}}$ (root control), and the accuracy when each query is placed at the midpoint of $\mathcal{T}_{\text{ref}}$. This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

additionally used $w = 8$ to counteract this effect and amplify the number of spaced-word matches. The corresponding results are presented in Fig. 3.12 for sequence coverages C from $\frac{1}{32}$ up to four in steps of powers of two. Three controls are presented to judge the quality of inferred placement positions: The accuracy of App-SpaM when executed with assembled reference sequences (assembled), the accuracy when every query is placed on the root of the tree (root control), and the accuracy when every query is placed on the midpoint of the tree (midpoint control). In general, the placement accuracy of App-SpaM is lower on unassembled reference sequences in all experiments compared to its accuracy on assembled sequences. For a coverage of four and $w = 12$ the ND increases by 14% for hiv-104 and by 15% for wol-43. For lower coverages, the accuracy discrepancy between assembled and unassembled sequences amplifies: For a coverage of one and $w = 12$ the ND is already 30% (hiv-104) and 31% (wol-43) higher than with assembled references. Since the average sequence length of hiv-104 is only 9 096 bp, the results deteriorate quickly for even lower coverages. When $C = \frac{1}{16}$ every reference genome is merely represented by a bag of reads containing 3.8 reads on average. In this case, App-SpaM is incapable of extracting useful placement information, and its proposed placements are only as good as when placing each query on the root straight

Table 3.3 – Runtime comparison of programs.

	App-SpaM		RAPPAS		APPLES	EPA-ng	EPA	pplacer
	$w = 12$	$w = 16$	$k = 6$	$k = 8$				
CPU-652								
preproc.	-	-	651	7 253	3 437	3 437	3 437	3 437
placement	152	79	710	454	2 804	1 315	194 338	9 257
total	152	79	1 361	7 707	6 241	4 752	197 775	12 694
CPU-512								
preproc.	-	-	1 070	12 144	1 879	1 879	1 879	1 879
placement	34	22	254	185	348	127	6 626	1 976
total	34	22	1 324	12 329	2 227	2 006	8 505	3 855

Comparison of runtimes for all programs on two data sets. 10 000 queries were placed for CPU-512 and 100 000 queries for CPU-652. All runtimes are denoted in seconds. For each data set, we present the time for preprocessing (*preproc.*), *placement*, and the *total* sum of preprocessing and placement with default parameters. The preprocessing steps include the generation of the query alignment and building the phylo-k-mer database for RAPPAS. This table is adapted from *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

away. Here, it is likely that no spaced-word matches are found for many query sequences and App-SpaM places such queries at the root anyway. For the same reason, no results are supplied for $C = \frac{1}{32}$ on the hiv-104 data set. In contrast, the average placement accuracy on wol-43 stays below the control methods for all sampling ratios.

The results of the runtime evaluation of all PP programs are outlined in Tab. 3.3. It shows the runtime of all programs in seconds (s) divided into the time requirements for preprocessing steps, the placement process itself, and the total sum of both. The preprocessing steps include the alignment of query sequences against the reference MSA or the construction of the phylo-k-mer database (pkDB) in RAPPAS. The preprocessing time is also declared for APPLES as it calculates query-reference distances based on the query alignments on default. There is no preprocessing step for App-SpaM. RAPPAS requires significant time to construct the pkDB; however, the database only must be built once for any reference data set. Then, multiple runs with different query reads may be performed consecutively without the need to rebuild the pkDB. The runtime of App-SpaM and RAPPAS is heavily dependent on the weight of the pattern or the size of the k-mers in the pkDB, respectively. Thus, for both programs, we report different runtimes with respect to these parameters. For all other programs, the runtimes are rather constant across all parameter combinations. The reported runtimes also depend on the utilized computing infrastructure and their absolute values carry little information. Instead, the relative in- or decrease of runtimes between the programs is of interest. App-SpaM requires the least time to place all query reads on both data sets. With default parameters, it requires 152 s on CPU-652 and 34 s on CPU-512. For comparison, the alignment-based programs EPA-ng and EPA require 4 752 s and 197 775 s, respectively. For EPA-ng, the runtime is primarily spent on the preprocessing step (3 437 s preprocessing, 1 315 s placement), while for EPA the total runtime is dominated by the placement (3 437 s preprocessing, 194 338 s placement). APPLES, the only other program that can perform alignment-free placement, requires 2 804 s to place the queries on CPU-652 and 348 s on

Table 3.4 – Memory usage of programs.

	App-SpaM		RAPPAS		APPLES	EPA-ng	EPA	pplacer
	$w = 12$	$w = 16$	$k = 6$	$k = 8$				
CPU-652								
preproc.	-	-	2 738	2 739	577	577	577	577
placement	253	235	1 606	2 513	561	669	5 575	995
max	253	235	2 738	2 739	557	669	5 575	995
CPU-512								
preproc.	-	-	3 305	3 602	122	122	122	122
placement	250	248	1 491	2 677	217	557	681	447
max	250	248	3 305	3 602	217	557	681	447

Comparison of peak memory usage for all PP programs on the CPU-652 and CPU-512 data sets. Memory usage is always indicated in MB and is taken from the column *PSS* of the results from the RES workflow of PEWO. This table is adapted from the Supplementary material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

CPU-512; this amounts to 18 times (CPU-652) and ten times (CPU-512) the runtime of App-SpaM. Thus, App-SpaM runs 30-60 times faster than the next fastest program EPA-ng (excluding RAPPAS with $k = 6$ since its accuracy deteriorates quickly) and three to five times faster than the placement step of RAPPAS.

The memory requirements for all programs are summarized in Tab. 3.4. All values are indicated in mega bytes (MB) and describe the peak memory usage of the programs measured as the *proportional set size* (PSS) by the RES workflow of PEWO. PSS is a reasonable estimate of the total peak memory requirement of a program where the memory usage of shared libraries is counted once. App-SpaM has the lowest memory requirement (253 MB) on CPU-652 and the third lowest (250 MB) on CPU-512 behind APPLES and pplacer. The pkDB of RAPPAS requires the largest amount of memory (2 739 MB on CPU-652 and 3 602 MB on CPU-512).

We also used App-SpaM to place up to 37 480 000 query reads onto the tara-3748 data set to demonstrate its placement capabilities. The results of these experiments are shown in Fig. 3.13. With parallel execution on 30 cores and default parameters, App-SpaM placed the 37 480 000 reads in 613 minutes. Increasing the pattern weight to $w = 16$ reduces the runtime to 475 minutes. The growth in runtime is linear with respect to the number of query sequences. This is expected since each query is placed individually on $\mathcal{T}_{\text{ref}}$ and does not influence previous or future placements. Thus, the computational load grows linearly with the number of query reads. No other program completed the placement of 37 480 000 on our system (note, however, that the used computing infrastructure does not comprise state-of-the-art components). A similar runtime study was conducted in the evaluation of EPA-ng [15].

3.1.3 Discussion

Phylogenetic placement (PP) is the task of integrating biological sequences of unknown origin into a known phylogeny of reference organisms to phylogenetically characterize the query sequences. The field of PP has experienced profound advancements during the last decade [334], including a variety of new software programs [14, 15, 332], a range of associated tools to

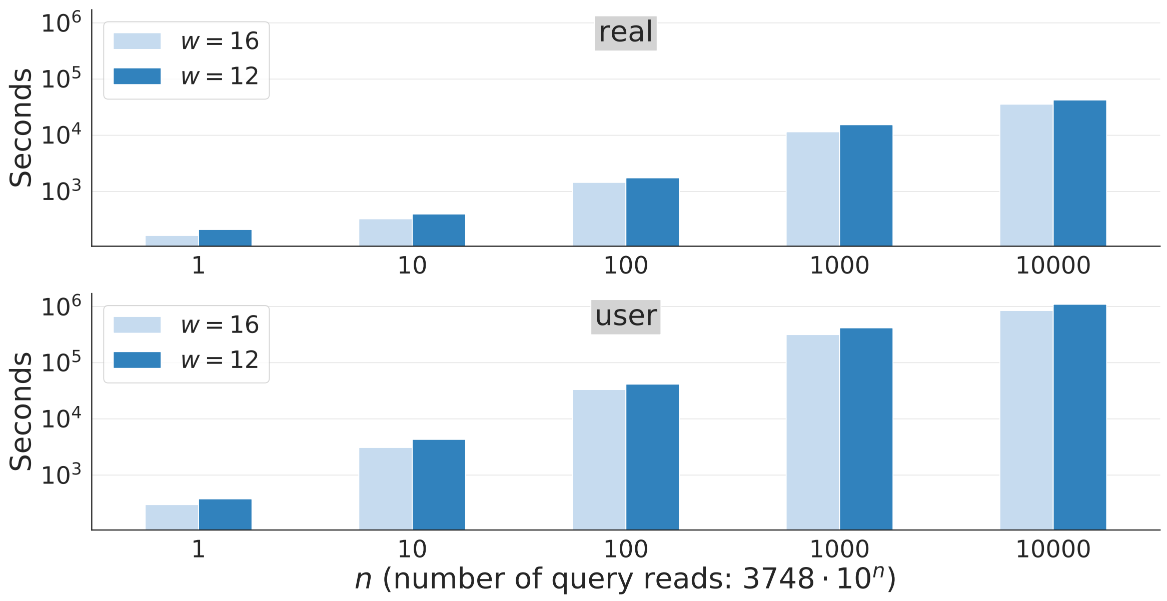


Figure 3.13 – Runtime of App-SpaM on the tara-3748 data set with large amounts of query reads. The runtime is shown in seconds (y-axes) with respect to the number of query reads (x-axes) for pattern weights of $w = 12$ and $w = 16$. Each bar represents the mean across two repeats. This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

handle phylogenetic placement data [333, 340, 357], and frameworks to evaluate and visualize the resulting placements [358]. PP has been predominantly applied in metataxonomics to identify short sequencing reads from metagenomic experiments. Its area of application has been limited mainly due to the nature of PP software: All previous programs for phylogenetic placement require a multiple sequence alignment of the reference sequences, and for most programs, the alignment is also required to include the query sequences.

We presented App-SpaM, a software program that performs phylogenetic placement without requiring reference or query alignments. The software is freely available via its Github page [359] and accessible as a Conda package on the Bioconda channel [360]. With App-SpaM, we pursue to continue the advancement of the domain of PP by introducing a novel alignment-free strategy to derive placement locations. Thereby, App-SpaM opens the possibility to process novel types of sequencing data for which phylogenetic placement was impossible previously. App-SpaM relies on spaced words that it extracts from all input sequences with respect to a predefined binary pattern. It then detects homologous sequence fragments, so-called spaced-word matches, between query and reference sequences using a filtering procedure as proposed in the program FSWM. However, App-SpaM is a novel implementation of this approach and is designed to perform phylogenetic placement for large amounts of sequencing data. We performed an extensive evaluation to assess the capabilities and limits of App-SpaM and the impact of different algorithmic designs on its placement accuracy. Furthermore, we conducted a thorough accuracy assessment, including five other programs for PP on a wide variety of metataxonomic data sets. To the best of our knowledge, no such comprehensive analysis for the evaluation of placement programs and their parameter settings has been performed previously. For this, we deployed the pruning-based accuracy workflow of the recently developed PEWO framework. We also extended PEWO to be able to simulate query reads with error profiles from common sequencing platforms by using the ART software and examined the quality of placements inferred with App-SpaM when only unassembled references are available; a scenario that is becoming more abundant, but cannot be handled natively by any other placement program other than App-SpaM.

When placing short query reads, App-SpaM is on par with other state-of-the-art ML-based programs, as indicated in Fig. 3.9. We measured the placement accuracy as the number of nodes in the reference tree between the proposed and expected placement branch, the so-called node distance (ND), and use both terms (*ND* and *accuracy*) interchangeably. App-SpaM's mean accuracy closely follows those of ML-based programs. In addition, it outperforms APPLES, the only other program that can perform reference-alignment-free placement. Note, however, that APPLES also uses query-reference alignments by default (and in our accuracy evaluation) to infer query-reference distances. App-SpaM has the best average performance of all programs in two of the eight inspected data sets. Furthermore, it is consistently on par with the other programs on the remaining data sets with the exception of hiv-104 and epa-714 where it does not quite reach the accuracy of the other programs. The variance of App-SpaM's placement accuracy across multiple pruning iterations is comparable to that of other programs. These results cover the metataxonomic setting (with the exception of the hiv-104 data set), the major field of application for phylogenetic placement [334, 361]. It is currently also the singular use case where all programs can be applied equally with respect to potential input data and without excessive runtimes. The program RAPPAS performs exceptionally well and reaches the lowest average ND overall and on four of the eight data sets specifically. RAPPAS is not based on the ML principle, but instead uses a pre-constructed database of k-mers that are presumed to be phylogenetically informing; it also does not require query alignments, but instead matches all k-mers of a query against the database to infer placement positions. As expected, all programs show an enhanced accuracy when longer query reads are used, see Fig. 3.10. This effect is stronger for ML-based programs and RAPPAS than for App-SpaM and APPLES.

App-SpaM relies on the *SpaM* approach where predefined patterns are utilized to find micro-alignments between query reads and reference distances. Our presented placement heuristics depend either on the *number* of spaced-word matches that pass the filtering step or on the estimated phylogenetic *distances* that are inferred by the Jukes-Cantor corrected average number of nucleotide substitutions at the don't care positions. This integral part of App-SpaM's algorithm implies its virtues but also causes troubles on the flip side. The advantages are evident: No time-intensive calculation of alignments is required, neither of the reference sequences nor of the query sequences. This allows App-SpaM to function in domains where it was impossible to apply phylogenetic placement previously, see Chpt. 5. In addition, App-SpaM is very fast compared to other PP programs, as demonstrated in Tab. 3.3; this applies even if the calculation of alignments is not taken into account. The spaced words are position independent and not affected by the presence of genomic rearrangements, gene duplications, gene deletions, or similar large-scale evolutionary effects. In general, they are a solid basis for the estimation of phylogenetic distances between whole genomes or bags of reads, as demonstrated in previous work [11, 166]. However, there are also drawbacks to consider when working with spaced words: First, our approach is not sensitive to short insertions or deletions, in contrast to alignment-based methods. The micro-alignments that are constituted by spaced-word matches have a fixed length and do not allow a shift of nucleotides in one of the two involved sequences. If an indel occurs at a sequence position App-SpaM most likely does not find spaced-word matches that include this position. If it still does, the match is found by pure chance because the nucleotides at the match positions on one side of the indel are identical by accident. Such a spurious spaced-word match is then subjected to the filtering procedure, where it is presumably filtered out, depending on the position of the indel within the spaced word. Thus, indels are unlikely to directly affect our distance estimates; instead, they lower the number of spaced-word matches because SpaMs are unable to model indels explicitly. For example, with a pattern length of $l = 44$, a single indel in the middle of a query sequence with 150 bp reduces the number of potential homologous spaced-word matches by roughly 41%. With each additional indel that is present in a query read, it becomes more improbable that *App-SpaM* finds a sufficient number of SpaMs to infer a well-founded placement position.

When using spaced-word matches, caution is also advised when low-complexity regions are present in the input sequences. We recommend end users to filter out all genomic segments of low complexity from the query and reference sequences, especially those regions that contain repeating sequence successions. This especially applies to eukaryotic sequences where such regions are more abundant than in prokaryotic genomes [362, 363]. The presence of repeats in a reference *or* query sequence results in a linear increase in spaced-word matches with respect to the length of the repeats. If they are present in reference *and* query sequences, the number of spaced-word matches grows quadratically. But not only does the runtime of App-SpaM suffer from repeats in the input sequences, but also the filtering procedure that detects homologous sequence positions becomes ineffective: Many spaced-word matches may pass the filtering step even if they do not originate from homologous sequence segments. This skews the number of spaced-word matches as well as the estimated substitution frequencies. Several programs exist to remove low-complexity regions [364, 365] and if such segments are suspected in the input sequences, they should be redacted prior to applying App-SpaM. We discuss the effect and handling of repetitive regions also in Sec. 6.1.

While App-SpaM's heuristics are extremely fast, they are also a simplification of the complex placement process and neglect considerable amounts of the available data. While ML-based programs infer near-optimal positions based on a detailed model of sequence evolution, App-SpaM simply performs placement based on those two references that have (or the single reference that has) the largest number of spaced-word matches to a query S_q. In contrast to APPLES, App-SpaM does not integrate the distances between S_q and *all* references into the designation of the placement branch or the assignment of branch lengths. It is striking that the accuracy of App-SpaM is on par with other placement programs nonetheless. This suggests that the gain in precision that is normally associated with ML-based methods is canceled out by the increase in the associated model complexity: The parameterization of the underlying evolutionary model and the construction of query alignments might counterbalance any gains in accuracy for ML-based programs and App-SpaM's lack of a sophisticated placement strategy is its curse and blessing at once. Furthermore, it is a surprising result that both heuristics that depend on estimated phylogenetic distances (Min-Dist and LCA-Dist) perform universally worse than those heuristics that solely depend on the number of spaced-word matches (LCA-Count and SpaM-X), see Fig. 3.4 and Fig. 3.5. The latter figure also indicates that Min-Dist and LCA-Dist get substantially worse with increasing pattern weights. We are unsure about the exact reasons for this behaviour. Again, one hypothesis is the aforementioned trade-off that comes with an increased model complexity: we suspect that the phylogenetic distances are not estimated reasonably well, especially under the presence of query sequences with short length. High weights reduce the number of spaced-word matches such that insufficient matches remain; see Fig. 3.14. Alternatively to SpaM-X, we also used a corresponding heuristic based on distance estimates instead of the number of spaced-word

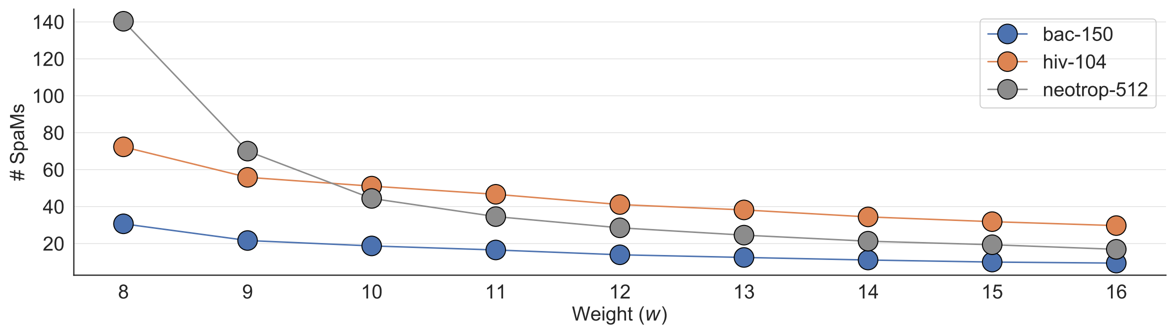


Figure 3.14 – Number of spaced-word matches with respect to the pattern weight $w \in \{8, 9, 10, 11, 12, 13, 14, 15, 16\}$ for three data sets.

matches; the results of this heuristic were equally poor as for MIN-DIST and LCA-DIST and, therefore, it is not shown in the evaluation. We discuss additional placement heuristics for APP-SPAM in Sec. 4.2.

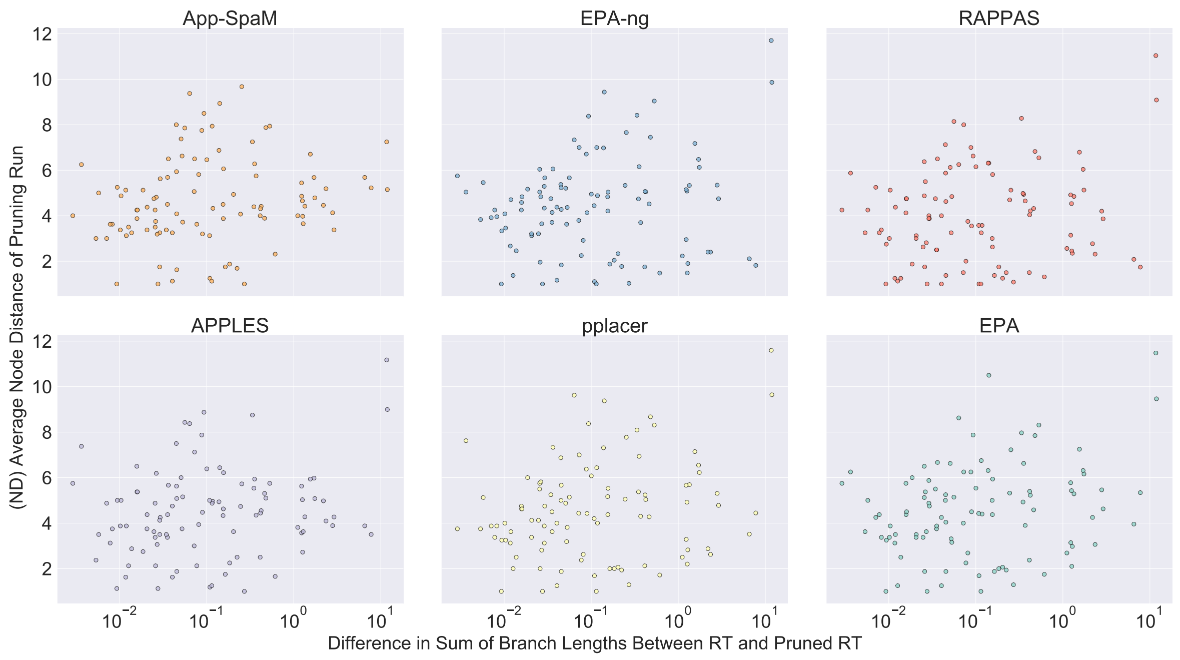
The current implementation of *App-SpaM* considers the reverse complement of sequences. This is essential, as the original strand of a sequence read by an NGS machine is not necessarily known. *App-SpaM* considers the references only in the singular direction of the supplied input data; but for each query S_q, spaced words are also extracted from its reverse complement in addition to its supplied direction. By this, we ensure that homologies between different sequence strands are taken into account. Admittedly, this strategy is not particularly memory efficient, since twice as many spaced words are stored for each query than necessary. Instead, an improved strategy that is regularly utilized in k-mer approaches is the use of *canonical* k-mers [148, 366]: The canonical version of any k-mer K is the lexicographically smaller string of K and its reverse complement. This concept can be directly transferred to spaced words: A canonical spaced word $W \in \Sigma \cup \{*\}$ is the lexicographically smaller string of W and its reverse complement, whereas the wildcard character $*$ is its own complement. Note that storing the reverse complement of W also requires to store the reverse complement of the associated don't care positions. The current strategy does *not* exacerbate memory issues in most cases though: The memory usage of APP-SPAM is dominated by two data structures whose sizes depend on the number and lengths of the input sequences. The first data structure keeps the two lists L_r and L_q of all extracted spaced words. Each spaced word, including all bases at the match and don't care positions, are saved in 28 bytes. Thus, the extraction of all spaced-words from a text file of sequences where every symbol is stored in a single bit requires 28 times the size of the file. This is solely relevant for the reference sequences since queries are processed in small batches and, consequently, do not cause excessive memory usage. Since queries are processed in batches, it is also not detrimental to store the reverse complement of each spaced-word explicitly with respect to the memory requirements. In addition to saving all spaced words, the second fundamental data structure stores the resulting statistics that are necessary to infer placement positions. This includes the number of spaced-word matches and the number of nucleotide mismatches between each query and reference sequence. Here, for m references and n queries per batch, the structure requires roughly $16 \cdot n \cdot m$ bytes. When using short reference sequences such as single marker genes, the second data structure dominates the memory footprint of APP-SPAM. This is also the case in the results of the RES workflow shown in Tab. 3.4. However, for long references the storage of the associated spaced-words dominates the overall memory usage. This is a hindrance in the broad application of APP-SPAM and we discuss strategies to overcome this limitation in Sec. 4.3

In previous work, it was suggested that using a set of multiple patterns improves the accuracy of alignment-free spaced-word methods over just using a single pattern [367]. Furthermore, it was discovered that a set of patterns that minimizes either the overlap complexity or the variance of the number of pattern-based matches reduces the variance of the accuracy between program runs and improved the overall accuracy of dependent methods [165]. With the results presented here, we cannot affirm that either of those two observations holds for App-SpaM: As shown in Fig. 3.7, increasing the number of patterns does not significantly alter the average accuracy, no matter which placement heuristic is employed. It appears that an increasing number of patterns restricts the ND variance over multiple program runs. However, this visual observation is not supported by statistical analyses: Using Levene's test [368] with a significance level $\alpha = 0.05$, there is no significant difference in the variance of the mean node distance between repetitions of the same pattern set size (compare Fig. 3.8), nor is there a significant difference in the variance of the mean accuracy across different pattern sizes (compare Fig. 3.7). There is also no significant difference between the average accuracy when using pattern sets of different sizes. Furthermore, the runtime grows linearly with the pattern set size; even if there was a significant drop in the ND variance, the small effect size would not compensate for the lengthened runtimes. Thus, like FSWM, App-SpaM uses a default pattern set size of one and we discourage the end user from increasing the number of used patterns to keep the runtime and memory requirements minimal. In general, App-SpaM exhibits excellent runtimes on the presented data sets and is ahead of the other programs by a wide margin; see Tab. 3.3. In the presented examples, the preprocessing of ML-based programs by itself takes longer than the overall time that App-SpaM requires for the whole placement. When preprocessing is excluded, the gap between App-SpaM and the more recent programs APPLES, RAPPAS, and EPA-ng shrinks, but the difference still remains at least an order of magnitude. The substantial reduction in runtime for EPA-ng compared to EPA also highlights the importance of improved heuristics for ML-based programs and the substantial progress that has been made in recent years.

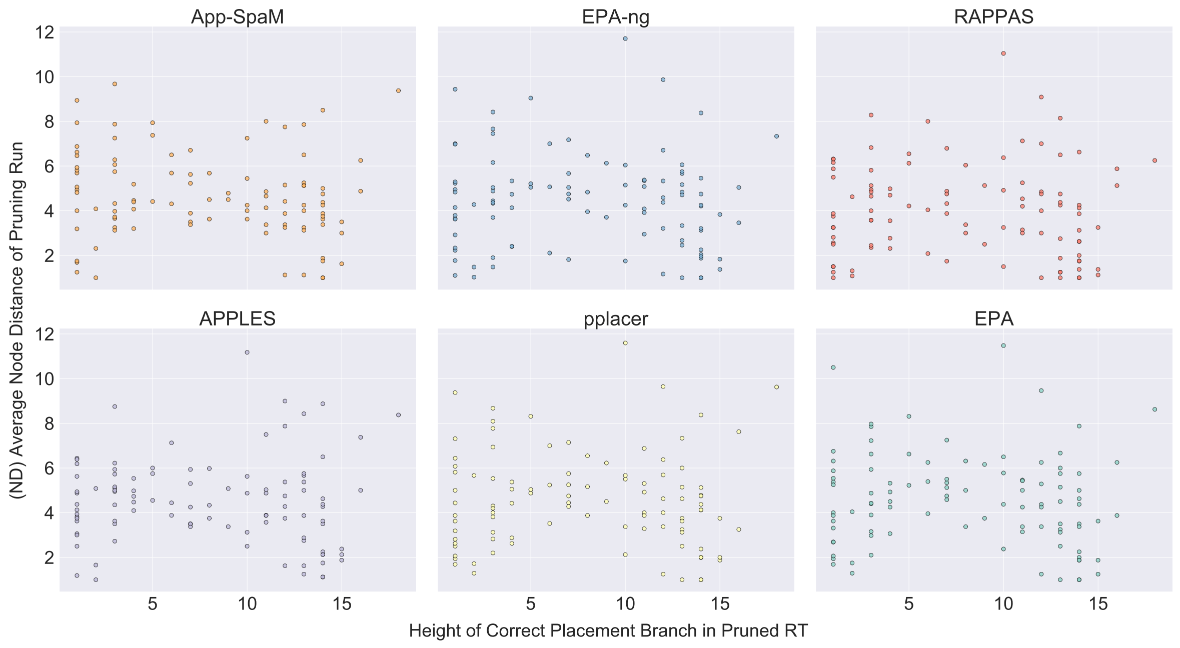
ML-based programs for PP assign multiple weighted placement locations to each single query S_q according to their normalized likelihood values. This is useful when the placement of S_q is uncertain and can guide decision making in subsequent analysis steps. Supplementary figures A.2 to A.12 show that the information carried by multiple weighted placement locations is superior for all ML-based programs than only considering their highest weighted placement: eND values are generally lower, indicating better performance than their respective ND values. Yet, the magnitude of this effect is negligible for most datasets. The software RAPPAS also reports weighted placement positions, even though it does not calculate likelihood values. Instead, it uses a measure associated with the logarithm of the product of the weighted phylo-k-mers on each placement edge to weight multiple placement branches. App-SpaM does not offer the possibility to specify multiple placement positions. However, different strategies to provide a measure of placement uncertainty for App-SpaM's placements are conceivable, see Sec. 4.4. The specification of placement uncertainty would constitute a beneficial feature for future versions of App-SpaM.

We extended the PEWO framework with the program ART to simulate sequencing reads of common sequencing platforms. The conspicuous results of this analysis are depicted in Fig. 3.11. PEWO performs query-to-reference alignments internally by using the hmm-align tool of the HMMER-suite [369]. All programs that rely on these query alignments show a severe drop in accuracy when ART-simulated reads are used. In this setting, they exhibit roughly double the node distance, as opposed to when reads are created by simply splitting the pruned references into shorter fragments. In contrast, the accuracy of App-SpaM barely changes, no matter how the provided query reads were generated. In general, we believe that App-SpaM is indeed robust to nucleotide substitution errors by design: Spaced words are an adaptive approach that is rarely affected by nucleotide mismatches compared to other alignment-free methods that rely on continuous k-mers. This is also supported by Fig. 3.11 which shows the small loss of accuracy when PEWO-generated reads are exchanged for ART-simulated reads. Contrarily, the simulated errors of ART might cause imprecise query alignments in PEWO's query alignment procedure. However, the reads generated by ART in our experiment follow a default error profile of the Illumina sequencing platforms and are expected to exhibit an average of approximately 0.0011 nucleotide substitutions per sequence position with 0.00009 insertions and 0.00011 deletions. For such low substitution and indel frequencies, it is not expected that the query alignments diverge notably. The results presented here are also in direct contradiction to the accuracy evaluation performed in RAPPAS where the loss of placement accuracy was examined in direct dependence on the indel and substitution rate of query reads. There, only little loss of accuracy was observed for the programs EPA, EPA-ng, pplacer, and RAPPAS. Despite thorough investigation, we were unable to determine the underlying reason for these discrepancies. They are likely to originate from the interaction of PEWO and ART in our implementation, which is freely accessible [353]. More careful tests are required to assess and interpret our results to determine their origin. For this, it would be beneficial if upcoming PEWO versions natively support the simulation of sequencing errors in the query reads according to common sequencing platforms, preferably with the option to adjust substitution and indel frequencies. Apart from this, the PEWO framework is a valuable tool to compare and assess PP programs, either on their own or among each other. Yet, another area of PEWO requires further attention: The PAC workflow uses the ND or eND as a measure of accuracy for PP programs. Both metrics are intrinsically coupled to the size of the reference tree and an identical ND has considerably different meanings for reference trees of varying sizes. For example, increasing the density of the taxon sampling in $\mathcal{T}_{\text{ref}}$ implies larger node distances even when queries are placed equally well in the same neighborhood of reference sequences. Thus, both metrics remain specific to singular reference trees and cannot be compared between different data sets. Normalizing the values of ND and eND by the total number of nodes in $\mathcal{T}_{\text{ref}}$ or the total size (sum of all branch lengths) of $\mathcal{T}_{\text{ref}}$ might alleviate some of these shortcomings. Furthermore, ND and eND do not take branch lengths into account, and additional metrics that incorporate the (normalized) sum of branch lengths between expected and proposed placement locations would be beneficial.

To assess the accuracy of all placement programs, PEWO performs multiple repetitions in each of which a random subtree from $\mathcal{T}_{\text{ref}}$ is pruned, a so-called pruning event. Pruning events are very disparate with respect to the size and topology of the remaining reference tree. The results in Fig. 3.9 suggest that certain pruning events are more challenging than others; in all data sets and for all PP programs, we observe outlier repetitions with a highly increased ND. Equally, we also note that the node distance approaches zero in some prunings. Hence, it seems evident that pruning events vary with respect to their 'difficulty' or 'complexity'; however, it remains unclear what features of a pruning event constitute its difficulty. We propose two properties of a pruning event that might be a proxy for its complexity. Let $s(\mathcal{T}_{\text{ref}})$ be the total sum of branch lengths of $\mathcal{T}_{\text{ref}}$. The first hypothesis is that the difference in the sum of branch lengths between $\mathcal{T}_{\text{ref}}$ and the pruned reference tree $\mathcal{T}_{\text{ref}}^{-}$ is an indication for pruning difficulty. We term this quantity $(s(\mathcal{T}_{\text{ref}}) - s(\mathcal{T}_{\text{ref}}^{-}))$ also the *size* of a pruning event. This difference is large if long branches or large clades are pruned from $\mathcal{T}_{\text{ref}}$. In the first case, the to-be-placed sequences are far away from their evolutionarily related next



(a) The average node distance (y-axes, log-scale) of each program (gray boxes) in dependence on the difference in the total sum of all branch lengths between $\mathcal{T}_{\text{ref}}$ and the pruned reference tree (x-axes).



(b) The average node distance (y-axes, log-scale) of each program (gray boxes) in dependence on the height of the correct placement branch in $\mathcal{T}_{\text{ref}}$ (x-axes).

Figure 3.15 – Relation between two proxies for the difficulty of an pruning event and the according placement accuracy. The upper part (a) shows the relation between the size of a pruning event and the placement accuracy for all six programs. The lower part (b) shows the relation between the location of the expected placement branch within $\mathcal{T}_{\text{ref}}$ and the placement accuracy for all six programs. This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

neighbors; in the latter case, a large amount of data is lost due to the pruning. Both cases could constitute a more complex placement scenario. In addition to the size of a pruning event, another indicator of pruning difficulty might be the location of the expected placement branch in the pruned tree $\mathcal{T}_{\text{ref}}^{-}$. We refer to the number of nodes on the longest path from the expected placement branch towards any leaf below as its *height*. The minimal height of 1 is reached when the expected placement branch is adjacent to a leaf node. The larger the height of the expected placement branch is, the more distant are the next reference sequences at the leaves and inferring secure placement branches might become more difficult. To assess whether one of these two properties is indeed an indicator of pruning difficulty, Fig. 3.15 illustrates the difficulty of pruning events with respect to the placement accuracy attained by each program on the BAC-150 data set. Supplementary Figure A.13 and Suppl. Fig. A.14 show additional results for NEOTROP-512 and BV-797, respectively. By visual inspection of Fig. 3.15a, no obvious trend is apparent: The accuracy for pruning events of all sizes varies considerably for all programs. Certain small pruning events have a large associated ND and, vice versa, large pruning events can exhibit small NDs. The same holds true for Fig. 3.15b which indicates that the height of the expected placement branch has no apparent impact on the ND of any program. We also calculated Spearman correlation coefficients between both proxies for pruning difficulty and the mean accuracy of all PP programs on the three data sets. When setting the significance level $\alpha = 0.05$, we report a significant correlation between the difference of sum of branch lengths and the ND for APP-SPAM on all three data sets. The same significant correlation also exists for all other programs on NEOTROP-512, for all programs except APPLES and RAPPAS on BV-797, and only for EPA on BAC-150. For the second proxy (height of expected placement branch), no significant correlations are present on the BAC-150 and NEOTROP-512 data sets; on BV-797 positive correlations are significant for all programs except APPLES and EPA. When considering these results, it has to be taken into account that larger prunings entail smaller reference trees, which in turn imply lower average node distances, no matter the relative quality of the placements. Given these results, the size of a pruning event indeed seems to be an adequate indication of the difficulty of a pruning event. Again, this shows that a proper sampling of reference organisms in $\mathcal{T}_{\text{ref}}$ is vital to ensure a high quality of placements.

APP-SPAM, as presented here, is already a full-fledged tool for performing fast alignment-free phylogenetic placement on diverse sequence data. However, we also observed some shortcomings of APP-SPAM with respect to its accuracy, the requirement of computational resources, and limitations with respect to specific properties of biological sequence data. In all these areas, there is promising potential to further improve APP-SPAM and we believe that it is worthwhile to address these issues to improve its capabilities. Nevertheless, APP-SPAM already presents a useful method to PP that offers a divergent strategy compared to previous approaches while providing similar accuracy and greatly accelerated runtimes.

Chapter 4

Improving Alignment-free Phylogenetic Placement

This chapter is derived from preliminary and subsequent work to the peer-reviewed open-access publication

Matthias Blanke and Burkhard Morgenstern.
"App-SpaM: Phylogenetic placement of short reads without sequence alignment."
In: *Bioinformatics Advances* (2021).

All text has been written exclusively for this thesis by Matthias Blanke, who also prepared all figures. Parts of Sec. 4.3 are derived from preliminary work by Christoph Elfmann (Department of Bioinformatics, Georg-August-Universität Göttingen) who applied iterative hash-based sampling to FSWM [370].

Using spaced words as the basis for phylogenetic placement involves multiple algorithmic decisions that influence the accuracy and speed of the placement process. During the design and implementation of App-SpaM we explored alternatives to the algorithmic procedure presented in Sec. 3.1. These alternatives include different placement heuristics, the use of evolutionary substitution models other than Jukes-Cantor, and the employment of spaced phylo-k-mers to perform a preselection of potential placement branches. We devised additional enhancements to App-SpaM after its publication; these enhancements include a sampling step for spaced words to minimize memory demands on large data sets and the computation of a measure of placement uncertainty. In the following, we revisit and evaluate all of App-SpaM's design decisions in detail.

4.1 Estimating Evolutionary Distances with Spaced Words

The evolutionary distance between two organisms is commonly represented by the number of nucleotide substitutions that occurred between their DNA sequences. Under the assumption of constant nucleotide substitution rates and independent sites, the number of substitutions between two sequences grows linearly with their evolutionary distance, as discussed in more detail in Subsec. 2.3.1. However, estimating the actual frequency of nucleotide substitutions is challenging in practice: The number of observed substitutions differs, potentially severely, from the true underlying frequency because occurrences of multiple mutations at the same sequence position are unidentifiable. Thus, a variety of models estimate the true number of nucleotide substitutions from the observed ones. Such models are parameterized by counting the observed substitutions in similar sequencing data, for example, from alignments; this prerequisite can often not be fulfilled when using alignment-free methods. In general, parameter-rich models tend to perform better when sufficient homogeneous data is available to estimate their parameters robustly. We refer to the true number of nucleotide substitutions per site between two sequences S_1 and S_2 as $d(S_1, S_2)$. For closely related sequences, the observed number of differences, also known as the Hamming distance or p-distance (see. Eq. 2.1), is a decent approximation for d. However, for larger evolutionary distances, the Hamming distance $p(S_1, S_2)$ consistently underestimates the true phylogenetic distance $d(S_1, S_2)$. Like *FSWM* and *Multi-SpaM*, *App-SpaM* uses the Jukes-Cantor (JC) model to estimate d from p. JC assumes that the base frequencies of all nucleotides are identical ($\pi_{\mathtt{A}} = \pi_{\mathtt{C}} = \pi_{\mathtt{G}} = \pi_{\mathtt{T}} = 0.25$) and that all nucleotide substitutions are of equal probability. Then, the distance between two sequences is estimated as

$$d(S_1, S_2) = -\frac{3}{4} \ln \left(1 - \frac{4}{3} \cdot p(S_1, S_2) \right) .$$

Using the Jukes-Cantor formula grants the advantage that only p must be estimated. Thus, App-SpaM simply counts the total number of nucleotide substitutions at the don't care positions and uses this single value as input for the JC correction. This not only provides efficient runtimes, but is also beneficial if no data is available to estimate parameters from.

We tested whether using the 2-parameter K80 model [46] instead of the Jukes-Cantor model influences the performance of *App-SpaM*. The K80 model distinguishes between transitions and transversions. Given a rate matrix Q such as in Eq. 2.5, the rate of all transitions are chosen to be equal ($b = e$) while another rate is specified for the transversions ($a = c = d = f$). For many DNA sequences, for example, in eukaryotic mitochondrial DNA [371], transitions are more frequent than transversion. If this is the case, the Jukes-Cantor formula underestimates d, unless the two involved sequences are closely related. We augmented App-SpaM with the possibility to distinguish between transitions and transversions when observing the don't care positions of those SpaMs that pass the filtering step. App-SpaM counts the number of transitions $q(S_q, S_r)$ and the number of transversions $r(S_q, S_r)$ between each query S_q and each reference S_r. Then, their phylogenetic distance is estimated as

$$d(S_q, S_r) = -\ln \left(\frac{1 - 2 \cdot q(S_q, S_r) - r(S_q, S_r)}{2} \right) + \ln \left(\frac{1 - 2 \cdot r(S_q, S_r)}{4} \right) . \tag{4.1}$$

The corresponding placement heuristics and branch length calculations are then carried out based on the K80-distances.

Using a different substitution model in App-SpaM has two effects: The direct influence of the chosen model on the distance estimates; and the indirect influence on the quality of inferred placement locations. The difference in the distance estimates between JC and K80 is exemplarily visualized in Fig. 4.1 for the hiv-104 data set. The figure shows distance estimates between three randomly selected query sequences and 30 randomly selected references. Additional results for the bac-150 and neotrop-512 data sets are shown from Suppl. Fig. B.1 to Suppl. Fig. B.4. We refer to the distance estimates of App-SpaM as d_{JC} and d_{K80}, depending on the employed model. All results show that $d_{\mathrm{JC}} < d_{\mathrm{K80}}$ consistently holds; thus, more transitions than transversions are present in the data sets. The deviation between d_{JC} and d_{K80} across all query-reference distances is minimal though: For more than 69% of all pairwise distance estimates in hiv-104, d_{JC} and d_{K80} differ less than 0.005. This ratio rises to over 86% for a difference less than 0.01. These results hold across all query-reference distances in all three data sets. In accordance with these results, we observe that the choice between these two substitution models has a negligible effect on the accuracy of the resulting phylogenetic placements. The ND metric remains identical regardless of which of the two models is used. Only the inferred branch lengths themselves differ slightly between the two models.

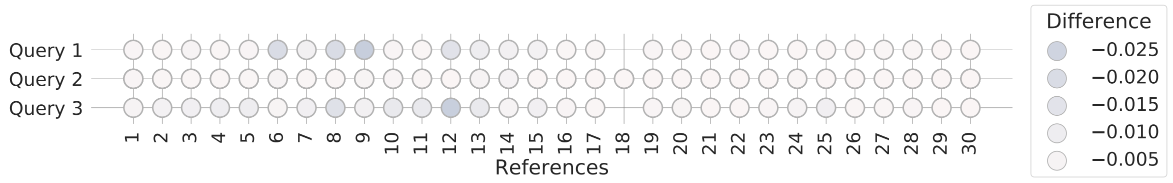
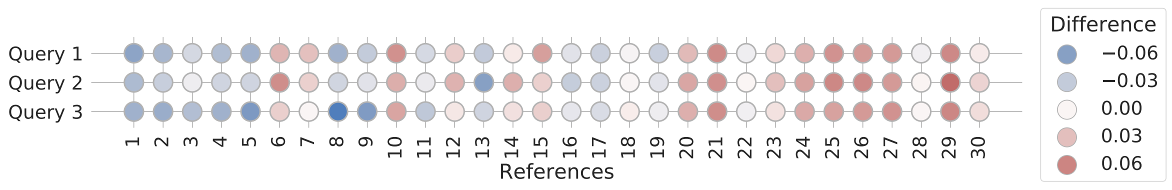


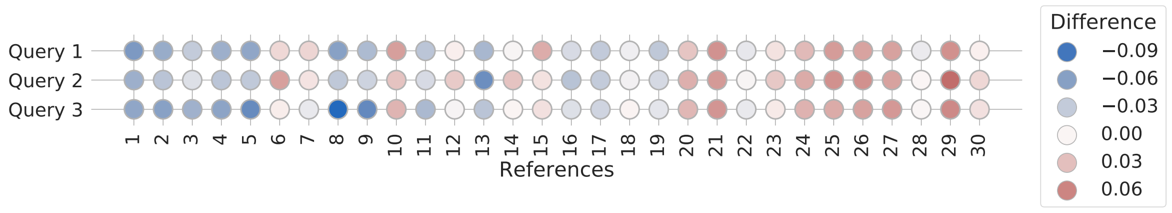
Figure 4.1 – Difference in estimates of the phylogenetic distance d depending on the substitution model when using App-SpaM on the hiv-104 data set. We denote the estimate of the Jukes-Cantor model as d_{JC} and the estimate of the K80 model as d_{K80}. Shown is $d_{\mathrm{JC}} - d_{\mathrm{K80}}$ (colored dots) in a heatmap for three randomly selected query sequences (y-axis) and 30 randomly selected reference distances (x-axis). Missing dots indicate that no spaced-word match was found between the respective query and reference.

While we observed that the relative difference between d_{JC} and d_{K80} is minimal and the selected model does not influence the placement accuracy, it remains unclear how robust App-SpaM's distance estimates are in the first place. Thus, we compared its distance estimates with those inferred from alignments. For this, we calculated pairwise Hamming distances p_{MSA} between all sequences in the reference alignment. As in App-SpaM, we ignored any undetermined sequence positions (typically marked with the character `N`) and gaps. We then applied either the JC correction to p_{MSA} (resulting in $d_{\mathrm{MSA(JC)}}$) or the K80 correction ($d_{\mathrm{MSA(K80)}}$). Following the pruning-based procedure of PEWO, we placed short reads from all pruned reference sequences and compared the distance estimates of App-SpaM averaged across all reads from the same reference to $d_{\mathrm{MSA(JC)}}$ or $d_{\mathrm{MSA(K80)}}$, depending on the model employed by App-SpaM. We performed five pruning events for each data set. Exemplary results from this procedure for three randomly selected queries and 30 randomly selected references are visualized in Fig. 4.2 for the hiv-104 data set. Note that each row represents the average distance of all query reads belonging to the same pruned reference sequence. The mean deviation between d_{JC} and $d_{\mathrm{MSA(JC)}}$ is only -0.005 across all repetitions on the hiv-104 data set. However, App-SpaM's distances deviate strongly between different query sequences from -0.06 to 0.06 for the JC model and from -0.09 to 0.06 for the K80 model. The variation in distance divergence is greater for the K80 model across all data sets. These results are consistent across all query-reference distances in the three data sets with the exception of certain sequences in the bac-150 data set; here, 25 out of the 150 references exhibit large evolutionary distances to all other queries with a mean distance larger than 0.5. All distance estimates that involve such references, or derived queries thereof, are severely underestimated compared to those distances inferred from alignments; see, for example, Suppl. Fig. B.2. Otherwise, the observations for the hiv-104 data set are consistent with observations on the bac-150 and neotrop-512 data set, see Suppl. Fig. B.1 to Suppl. Fig. B.4.

When $d_{\mathrm{MSA(JC)}}$ and $d_{\mathrm{MSA(K80)}}$ are assumed to be robust estimates of the true distance d, the presented results indicate that the distance estimation for short reads using the filtered-spaced words technique is challenging. The program Read-SpaM demonstrates that distances between two *bags of reads* from two genomes are accurately estimated using the spaced-word matches approach. In contrast, App-SpaM estimates distances to *single* read sequences with a length of only 150 bp in the presented experiments. Although a deviation from the true distances is expected for short queries, we did not expect the pronounced discrepancy of average distances between all query sequences derived from a single reference, as observed in Fig. 4.2. However, even the total number of queries originating from a single pruned reference often stays below ten when the reference comprises a single marker gene. In comparison,



(a) The difference $d_{\text{MSA(JC)}} - d_{\text{JC}}$ for the distance estimates of the Jukes-Cantor model.



(b) The difference $d_{\text{MSA(K80)}} - d_{\text{K80}}$ for the distance estimates of the Kimura model.

Figure 4.2 – Divergence of App-SpaM's phylogenetic distance estimates from the 'true' distance on the hiv-104 data set. The true distances were calculated from multiple sequence alignments as the average number of nucleotide substitutions per sequence position, corrected with the Jukes-Cantor or K80 formula. We denote App-SpaM's estimates d_{JC} and d_{K80} for the Jukes-Cantor and Kimura 2 parameter model, respectively. Shown is the difference between those estimates and true distances (colored dots) in heatmaps for three randomly selected query sequences (y-axis) and 30 randomly selected reference distances (x-axis). Missing dots indicate that no spaced-word match was found between the respective query and reference.

Read-SpaM's bags of reads evaluation had orders of magnitude more reads available, even when using genomes of low coverage.

The absolute difference between App-SpaM's distance estimates and d_{MSA} is not the only decisive factor for inferring placement positions with distance-based heuristics. Many alignment-free programs for phylogeny reconstruction estimate pairwise sequence distances that bear no biological meaning after all; however, their values are consistent among each other, thus allowing the reconstruction of topologically correct phylogenetic trees. Similarly, App-SpaM's estimated p-distances might not coincide with d_{MSA}, but could be consistent among each other. For this, we calculated Pearson's correlation coefficient between d_{JC} and $d_{\text{MSA(JC)}}$ for query-reference distances on the hiv-104, bac-150, and neotrop-512 data sets. Across all distances in five pruning events, the correlation coefficient is 0.352 for bac-150, 0.877 for hiv-104, and 0.372 for neotrop-512. Interestingly, the highest correlation by far occurs for hiv-104, the data set for which App-SpaM's results are inferior to those of the alignment-based programs as visualized in Fig. 3.9. Apparently, App-SpaM has good placement results despite its rather unstable distance estimates. When moving from the JC model to the K80 model, we also observed that the variance of distance estimates increased across all experiments. This was to be expected since estimating both transitions and transversions instead of only substitutions from the limited number of available base pairs is challenging. Therefore, we assume that the use of even more parameter-rich models such as the GTR model is unlikely to improve App-SpaM's distance estimates. Similarly, we do not expect that the use of such models would improve our placement results. The presented findings are also indicative of why distance-based heuristics perform worse than those heuristics that are based on the number of spaced-word matches; the latter metric might be a better indicator of sequence similarity after all. Furthermore, the discrepancy between App-SpaM's p-distances and p_{MSA} indicates that spurious spaced-word matches are mistakenly involved in the distance calculation. However, both propositions require a closer observation of the number and distribution of spaced-word matches between query and reference sequences, all of which depend on the filtering procedure.

In addition to the implemented substitution models, the matrix M that specifies the substitution scores for the filtering procedure as well as the filtering threshold t are other components that could influence the precision of App-SpaM. Both have a direct impact on the number and composition of spaced-word matches that are considered in all subsequent steps. App-SpaM uses the HOXD70 matrix as a default for M which is defined as

$$M = M_{\text{HOXD70}} = \begin{pmatrix} 91 & -114 & -31 & -123 \\ -114 & 100 & -125 & -31 \\ -31 & -125 & 100 & -114 \\ -123 & -31 & -114 & 91 \end{pmatrix} \tag{4.2}$$

whereas every entry M_{ij} represents the substitution score for the symbols σ_i and σ_j in $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. HOXD70 was originally designed to score large alignments between the genomes of human and mouse [345]. As such, it is not evident whether it also works well for sequences of other origin and type, such as short marker genes present in metataxonomics. The main purpose of M is to separate random background matches from valuable matches that originate from homologous sequence positions. For this, every spaced-word match is assigned a *score* with respect to M that is calculated as the sum of substitution scores of corresponding nucleotides at every don't care position. The two score distributions of background and homologous matches are clearly separated when complete genomes are compared [167]. The heights of both distributions and their overlap depend on the similarity and length of the two compared sequences. The filtering procedure uses the threshold $t = 0$ to remove all matches with a score below t. In the optimal case, the distributions of background and homologous matches have very little overlap and t separates them appropriately.

Spaced-word histograms, also referred to as *spamograms*, are a useful technique to visualize the distribution of homologous and background matches. Given a set of spaced-word matches, a spamogram displays how the matches are distributed with respect to their score. This provides information on if and how well the score distributions of background and homologous matches are separated. We examined the distribution of spaced words with respect to their score for different data sets to analyze whether the default parameters of t and M are chosen appropriately for App-SpaM. In addition to the HOXD70 substitution matrix, we assessed how the simple binary matrix

$$M = M_{\text{bin}} = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix} \tag{4.3}$$

affects spamograms and whether it improves the differentiation between homologous and background matches. For this, we recorded all spaced-word matches together with the sequences they originated from when performing placement on the bac-150 and wol-43 data sets.

For FSWM, a spamogram contains the complete set of spaced-word matches between two genomes. In contrast, a single run of App-SpaM generates a separate set of spaced-word matches for each query-reference pair. It is beneficial to plot each of these sets as a single spamogram. Figure 4.3 displays a typical example of a spamogram for the wol-43 data set

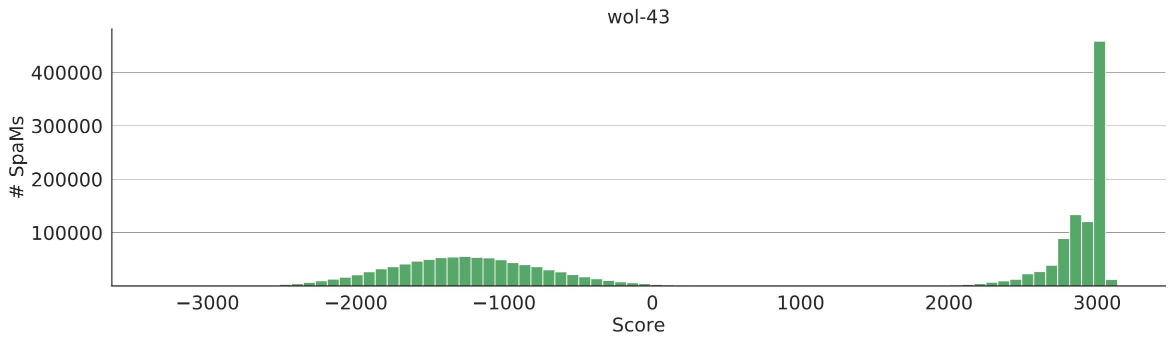


Figure 4.3 – A spaced-word histogram for SpaMs between all queries and a single reference. The distribution of background matches (all SpaMs on the left with negative score) are clearly separated from the homologous matches on the right.

where references comprise whole genomes. The spamogram comprises matches between all queries of a single run of App-SpaM to a *single* reference sequence with length 52 768 066 bp. One such spamogram exists for each reference sequence in $\mathcal{T}_{\text{ref}}$. These spamograms resemble what was observed in FSWM and Prot-SpaM: two clearly separated peaks for the homologous and background matches exist. However, the shape of the homologous peak is dubious. It does not follow a normal distribution as observed in Prot-SpaM; instead, it has multiple smaller peaks, for example, at a score of 2 800 and at a score of 3 000. This is due to the varying similarities of the short query reads depending on their region of origin in the pruned reference sequence. Accordingly, Suppl. Fig. B.11 shows a spamogram between *all* SpaMs that occur during a program run of App-SpaM. Several millions of spaced-word matches arise between the input sequences and have to be processed. The height of the background peak depends predominantly on the pattern weight w; a lower weight causes higher background peaks. Additional spamograms between a *single* query read of length 150 bp and a *single* reference and between a *single* query and all references are provided in Suppl. Fig. B.12 and Suppl. Fig. B.13, respectively. For a single query read, there are roughly 160 homologous spaced-word matches on which the inferred placement position is based. When all references are taken into account, the homologous peak suggests a larger variance of scores.

Figure 4.4 displays multiple spamograms between a single query S_q and different references for both the HOXD70 and the binary substitution matrix on the wol-43 data set. The augmented density estimates allow for a direct assessment of the similarity between S_q and each reference: references with high-scoring homologous peaks are assumed to have very similar sequence fragments to S_q while those references with lower average scores are more disparate. The number of homologous spaced-word matches in each spamogram is 160 on average, as indicated in Suppl. Fig. B.12. Some references do not have a homologous peak. There is little difference between the overall form of the spamograms for the different substitution matrices. In both cases, the background distribution extends slightly above a score of zero. All of these results are in accordance with previous findings of other SpaM-approaches: In the presence of long reference sequences, there are many spaced words for each query sequence. The SpaMs split into a background peak and a homologous peak which are clearly separated from one another. There is an apparent need to remove SpaMs with low scores, as they likely occurred by pure chance. A filtering threshold of $t = 0$ or slightly higher is an appropriate choice to separate the SpaMs accordingly.

However, the spamograms for data sets that comprise short marker genes deviate substan-

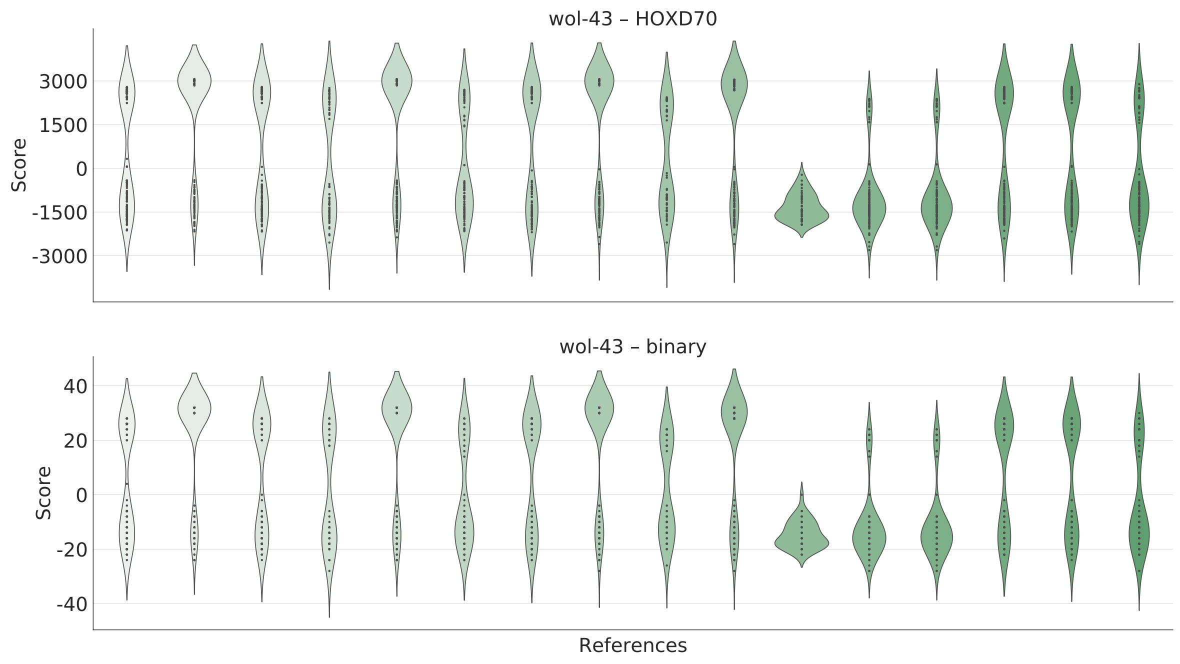


Figure 4.4 – The number of spaced-word matches with a specific score (y-axis) between a single query S_q to a set of randomly chosen references (x-axis) using either (a) the HOXD70 matrix or (b) a binary matrix of substitution scores. Each violin plot (green patch) is a density estimate of a single spamogram.

tially from those of WOL-43. Figure 4.5 shows spamograms between a single query and 20 references on the BAC-150 data set. Here, most spamograms exhibit only a single homologous peak and have no matches with a score below zero, apart from a few exceptions. The reason for this is that the references are short marker genes; in this scenario the vast majority of all matches are homologous and random matches are rare. Furthermore, most references provide a homologous segment for each query, resulting in large homologous peaks. Supplementary Figure B.7 shows the distribution of all spaced-word matches from a single run of App-SpaM on the BAC-150 data set. Here, the distributions of background and homologous matches are overlapping. This is mainly due to the specific properties of the BAC-150 data set, which has a small portion of disparate reference sequences with strongly divergent evolutionary relationships. Thus, considering *all* SpaMs at the same time implies that very different scores occur among query and reference sequences. This is resolved when singular spamograms are plotted for each query or reference, respectively. Again, using a smaller pattern weight causes large amounts of background matches, even for the metataxonomic data sets as demonstrated in Suppl. Fig. B.8. This observation is supported by Suppl. Fig. B.9 and Suppl. Fig. B.10, both of which display results for the data sets HIV-104 and NEOTROP-512 for pattern weights of $w = 12$ and $w = 8$.

The filtering procedure is an essential step in order to remove spaced-word matches that occurred purely by chance. These *background* matches skew the estimate of the number of homologous matches and of inferred query-reference distances. Therefore, choosing an appropriate substitution matrix M and filtering threshold t is critical as they determine which matches are kept and which are discarded. In all presented examples, we observe that homologous matches are clearly separated from background matches when a single query sequence is considered. This finding is independent of the choice of the substitution matrix

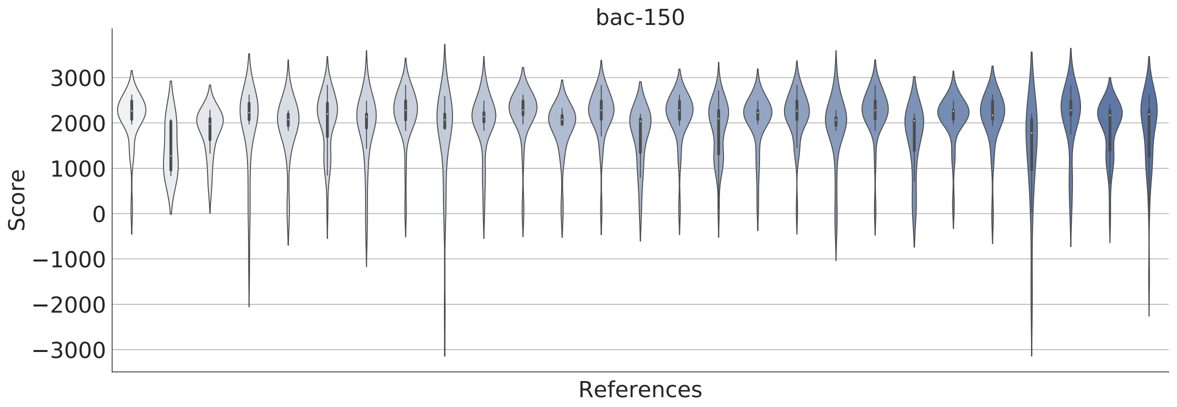


Figure 4.5 – The number of spaced-word matches with a specific score (y-axis) between a single query S_q to a set of randomly chosen references (x-axis) using the default HOXD70 substitution matrix. Each violin plot (blue patch) is a density estimate of a single spamogram.

M, as it has little influence on the overall form of distribution of SpaM-scores. The number of SpaMs varies greatly and depends on the length of the input sequences. Longer input sequences entail larger numbers of background matches, as does lowering the pattern weight. When App-SpaM is used for metataxonomic data with its default weight $w = 12$, the filtering procedure is not necessarily required since there are only a few background SpaMs at most. However, when lowering w or using longer input sequences, the filtering procedure becomes indispensable in order to remove random SpaMs from further consideration. The presented spaced-word histograms suggest that the default threshold $t = 0$ is slightly too low as the distribution of background matches seems to exceed zero regularly. It might be beneficial to increase t, for example $t = 500$, to ensure that all background matches are discarded. According to our results, raising the threshold does not cut off the homologous distribution in most cases, apart from the diverse bac-150 data set. In general, choosing an appropriate threshold becomes more difficult, as the reference sequences exhibit a diverse range of distances to the queries. The homologous distribution shifts towards a lower mean for distantly related references and can overlap with the background distribution, see Suppl. Fig. B.7. App-SpaM would benefit from a dynamic adjustment of t that is determined with respect to the observed overlap between the distributions. Fixing t at any value always results in a trade-off in terms of sensitivity and specificity for the selection of homologous SpaMs.

4.2 Placement Heuristics

App-SpaM performs phylogenetic placement in a two-step procedure: First, it finds all spaced-word matches between every query read and reference sequence, removes those that have a score below the threshold t, and uses the remaining ones to calculate the number of SpaMs s and the phylogenetic distance d as discussed in detail previously. Second, App-SpaM uses one of several placement heuristics to insert a query sequence into $\mathcal{T}_{\text{ref}}$ based on *either* the number of SpaMs remaining after filtering *or* the estimated phylogenetic distance to each reference sequence. However, both sources of information are not combined in a single heuristic. In this section, we present multiple additional heuristics for placement that extend upon the principles introduced in Sec. 3.1. We assume that a query sequence S_q is placed onto the phylogenetic tree $\mathcal{T}_{\text{ref}}$ comprising m leaves labeled with m reference sequences $\{S_1, S_2, \ldots, S_m\}$. For this, an existing *placement* edge e_q is split into two parts by inserting a new node n_q. The proximal part of the divided edge is called e_{q1} and the distal part e_{q2}. Then a new leaf node n'_q is added to $\mathcal{T}_{\text{ref}}$ which is connected to n_q via a new edge e'_q. Branch lengths must be specified for e_{q1}, e_{q2}, and e'_q, whereas the restriction $l(e_{q1}) + l(e_{q2}) = l(e_q)$ is imposed. The resulting placement tree is denoted as $\mathcal{T}_q$. To select a suitable placement edge e_q we use the information calculated between S_q and all references represented by the leaves in $\mathcal{T}_{\text{ref}}$: For each reference sequence S_r at each leaf l_r, $r \in [m]$ we know the number $s(S_q, S_r)$ of filtered SpaMs as well as the calculated phylogenetic distance $d(S_q, S_r)$.

Methods

To determine a placement branch, the first group of heuristics traverses $\mathcal{T}_{\text{ref}}$ from its root towards its leaves following the branches along an appropriate path with respect to the leaf annotations s and d. The traversal stops at a suitable branch e_q where S_q gets inserted. For this, we introduce several measures for the internal nodes of $\mathcal{T}_{\text{ref}}$ derived from the spaced-word statistics at the leaves. For an inner node u of the rooted reference tree, we denote the set of all references in the clade induced by u as $\mathcal{S}_u = \{S_r \mid S_r \text{ is a leaf in the subtree induced by } u\}$. Furthermore, we define the three measures

$$s_{\text{avg}}(S_q, \mathcal{S}_u) = \frac{1}{|\mathcal{S}_u|} \cdot \sum_{S_r \in \mathcal{S}_u} s(S_q, S_r)\,, \tag{4.4}$$

$$s_{\text{sum}}(S_q, \mathcal{S}_u) = \sum_{S_r \in \mathcal{S}_u} s(S_q, S_r)\,, \tag{4.5}$$

$$s_{\text{max}}(S_q, \mathcal{S}_u) = \max_{S_r \in \mathcal{S}_u} s(S_q, S_r)\,. \tag{4.6}$$

Thus, $s_{\text{avg}}(S_q, \mathcal{S}_u)$ is the average number of SpaMs of each reference below u, $s_{\text{sum}}(S_q, \mathcal{S}_u)$ is the total number of SpaMs across all references below u, and $s_{\text{max}}(S_q, \mathcal{S}_u)$ is the maximal number of spaced words of any reference below u. Similarly, we define for the Jukes-Cantor corrected distances the two measures

$$d_{\text{avg}}(S_q, \mathcal{S}_u) = \frac{1}{|\mathcal{S}_u|} \cdot \sum_{S_r \in \mathcal{S}_u} d(S_q, S_r)\,, \tag{4.7}$$

$$d_{\text{min}}(S_q, \mathcal{S}_u) = \min_{S_r \in \mathcal{S}_u} d(S_q, S_r)\,. \tag{4.8}$$

We traverse $\mathcal{T}_{\text{ref}}$ starting at its root; since it is strictly bifurcating, every internal node has two descendant nodes u and v. Let $s_\star(\cdot\,,\cdot)$ denote one of the three measures from Eq. 4.4 to Eq. 4.6. At each internal node, we evaluate whether $s_\star(S_q, \mathcal{S}_u) > s_\star(S_q, \mathcal{S}_v)$ holds; if so, the search is continued at node u, otherwise at node v. For Eq. 4.7 and Eq. 4.8 the search is continued at node u if $d_\star(S_q, \mathcal{S}_u) < d_\star(S_q, \mathcal{S}_v)$. Thus, the search progresses from the root towards the leaves along a path that has either many SpaMs or low distance estimates. This procedure would progress until a single leaf node is reached. For s_{max} and d_{max}, the resulting heuristics would then be identical to SPAM-COUNT and MIN-DIST, respectively. However, we use a *stopping criterion* that potentially halts the search before a leaf node is reached. Here, the idea is to stop at an inner node for which $s_\star(S_q, \mathcal{S}_u)$ and $s_\star(S_q, \mathcal{S}_v)$ (or $d_\star(S_q, \mathcal{S}_u)$ and $d_\star(S_q, \mathcal{S}_v)$) are similar to each other and both subtrees induced by u and v provide plausible placement positions. In such instances, it is not possible to reach an informed decision about which subtree is 'better' for the placement of S_q. We propose two stopping criteria: The first one evaluates whether

$$\otimes_1 = |s_\star(S_q, \mathcal{S}_u) - s_\star(S_q, \mathcal{S}_v)| > \frac{s_\star(S_q, \mathcal{S}_u) + s_\star(S_q, \mathcal{S}_v)}{r} \tag{4.9}$$

holds. The $\otimes_1$ criterion is parameterized by r for which we use values of $r \in \{2, 5, 10\}$. We also use this stopping criterion for $d(\cdot\,, \cdot)$. The second stopping criterion is defined as

$$\otimes_2 = |s_\star(S_q, \mathcal{S}_u) - s_\star(S_q, \mathcal{S}_v)| > s_{\text{sum}}(S_q, \mathcal{S}_{\text{root}}) \cdot r\,. \tag{4.10}$$

This criterion considers the difference between $s_\star(S_q, \mathcal{S}_u)$ and $s_\star(S_q, \mathcal{S}_v)$ depending on the total number of spaced-word matches across all reference sequences. Here, we use values of $r \in \{0.01, 0.05, 0.1\}$.

Another group of placement heuristics chooses the branch above the LCA of *a set* of references that have an exceptionally high number of spaced-word matches or particularly low distance estimates to S_q. For this, we consider the list of SpaM-counts $\mathbf{s} = [s(S_q, S_1), s(S_q, S_2), \ldots, s(S_q, S_m)]$ between S_q and each reference sequence. The first quartile Q_1 is the median of the lower half of all values in $\mathbf{s}$ and Q_3 the median of the upper half of the values in $\mathbf{s}$. The interquartile range is defined as $\text{IQR} = Q_3 - Q_1$. We determine a threshold $b = Q_3 + r \cdot \text{IQR}$ that is parameterized by r. The first heuristic determines the set $\mathcal{S}_b = \{S_r \mid s(S_q, S_r) \geq b\}$ and chooses the branch above the LCA of all references in $\mathcal{S}_b$ to be the placement branch e_q. We test this heuristic for the parameter values $r \in \{1.1, 1.3, 1.5\}$. The second heuristic is based on the distance estimates and chooses the branch above the LCA of $\{S_r \mid d(S_q, S_r) \leq b\}$ for $b = Q_1 - r \cdot \text{IQR}$; again, we test $r \in \{1.1, 1.3, 1.5\}$ for this heuristic.

As established earlier, the distance estimates of single query reads inferred with App-SpaM are not accurate. Thus, it is questionable whether they are suited for finding appropriate placement positions. Both originally proposed heuristics Min-Dist and LCA-Dist were inferior to SpaM-based heuristics and their accuracy varied considerably with respect to other parameters such as the pattern weight. Still, our distance estimates showed significant correlation with distances estimated from alignments, although the effect size varied strongly between different data sets. Combining the distance measure d with the number of spaced-word matches s might overcome the limitations we noticed when only using d. We combine the leaf annotations d and s by assigning each leaf a score m defined as $m = \frac{d}{s}$: The distance annotation at each leaf is weighted by the inverse of the number of spaced-word matches for the same reference sequence. Leaves with a low distance and many spaced-word matches receive a low score and are likely close relatives. On the contrary, it is improbable that leaves with only a few spaced-word matches are closely related to S_q. We define three heuristics based on this measure that follow our previous approaches: The first heuristic simply chooses the branch above the leaf with the highest score. The second heuristic chooses the branch above the LCA of the two leaves with highest score. The third heuristic uses a tree descent from the root towards the leaves as described above and follows the path along the internal nodes according to Eq. 4.7 and Eq. 4.8.

Results

We evaluated all above heuristics with different parameters by running PEWO's pruning based accuracy evaluation workflow on the five data sets bac-150, hiv-104, neotrop-512, bv-797, and epa-628 with 50 iterations each. We recorded the average node distance, subsequently also referred to as accuracy, of each heuristic across all iterations. Table 4.1

Table 4.1 – Additional placement heuristics of App-SpaM.

name	SpaMs	Dist	$m(\cdot\,,\cdot)$	Stopping Criterion	r
TD-SpaM-Count-Max	✓		$s_{\max}(S_q, \mathcal{S}_u)$	$\otimes_1$	$\{2, 5, 10\}$
TD-SpaM-Count-Avg	✓		$s_{\text{avg}}(S_q, \mathcal{S}_u)$	$\otimes_1$	$\{2, 5, 10\}$
TD-SpaM-Count-Sum	✓		$s_{\text{sum}}(S_q, \mathcal{S}_u)$	$\otimes_1$	$\{2, 5, 10\}$
TD-SpaM-Count-Max-Alt	✓		$s_{\max}(S_q, \mathcal{S}_u)$	$\otimes_2$	$\{2, 5, 10\}$
TD-SpaM-Count-Avg-Alt	✓		$s_{\text{avg}}(S_q, \mathcal{S}_u)$	$\otimes_2$	$\{2, 5, 10\}$
TD-SpaM-Count-Sum-Alt	✓		$s_{\text{sum}}(S_q, \mathcal{S}_u)$	$\otimes_2$	$\{2, 5, 10\}$
TD-Dist-Min		✓	$d_{\min}(S_q, \mathcal{S}_u)$	$\otimes_1$	$\{2, 5, 10\}$
TD-Dist-Avg		✓	$d_{\text{avg}}(S_q, \mathcal{S}_u)$	$\otimes_1$	$\{2, 5, 10\}$
LCA-IQR-SpaM-Count	✓				$\{.01, .05, .1\}$
LCA-IQR-Dist		✓			$\{.01, .05, .1\}$
Best-Score	✓	✓			
LCA-Score	✓	✓			
TD-Score-Max	✓	✓			$\{.01, .05, .1\}$

Additional placement heuristics for App-SpaM. In their order of appearance, the columns present the *name* of the placement heuristic, whether they utilize the number of spaced-words (*SpaMs*), whether they utilize the distance estimates (*Dist*), with which formula $s_\star(\cdot\,,\cdot)$ or $d_\star(\cdot\,,\cdot)$ the tree traversal is performed, which stopping criterion they rely on, and the parameters they are tested with. TD is the short form for Tree-Descent. When referring to a parameterized version of a heuristic, the parameter is appended to its name. For example, TD-SpaM-Count-Max-2 refers to the TD-SpaM-Count-Max heuristic with $r = 2$.

gives an overview of all heuristics, their properties, and designated names that are used in the following. Fig. 4.6 displays all newly introduced heuristics that depend solely on the number of spaced-word matches. In addition, it shows the accuracy of the default heuristic SpaM-X, the LCA-Count heuristic, and the accuracy of the program EPA-ng with default parameters. Accordingly, Fig. 4.7 displays the accuracy of all other newly introduced heuristics that depend solely on the distance estimates or on both the number of spaced-word matches and distance estimates. No single heuristic performs best on all five data sets. Certain heuristics perform well on a subset of data sets: The TD-SpaM-Count-Sum-2 performs a tree descent along the path that has the highest overall number of spaced-word matches and stops according to $\otimes_1$. This heuristic performs best in three of the five data sets with statistically significant improvements over other heuristics in the data sets bac-150 and hiv-104 ($P < .01$ in more than 90% of cases, otherwise $P < .05$). However, it performs slightly worse than App-SpaM's default heuristic on the bv-797 data set and substantially worse on the epa-628 data set. On the contrary, the TD-SpaM-Count-Max-Alt heuristic performs well on the data sets bv-797 and epa-628, but is sub-par on the other three data sets. The LCA-Score heuristic that uses both the number of spaced-word matches and estimated distances is close to optimal on the hiv-104 and neotrop-512 data sets but not as accurate on the other three. Some of the tested heuristics produce poor placement results on all data sets, such as the TD-SpaM-Count-Sum-Alt heuristic for all three parameter values of r, and most of the TD-Dist-heuristics. The performance of the TD-SpaM-Count-Avg heuristic varies greatly across all data sets. Here, we suspect the cause to be the averaging of the SpaM-counts across all leaves in both subtrees. The resulting average values depend predominantly on the topology of the tree itself. For example, if the reference sequence with

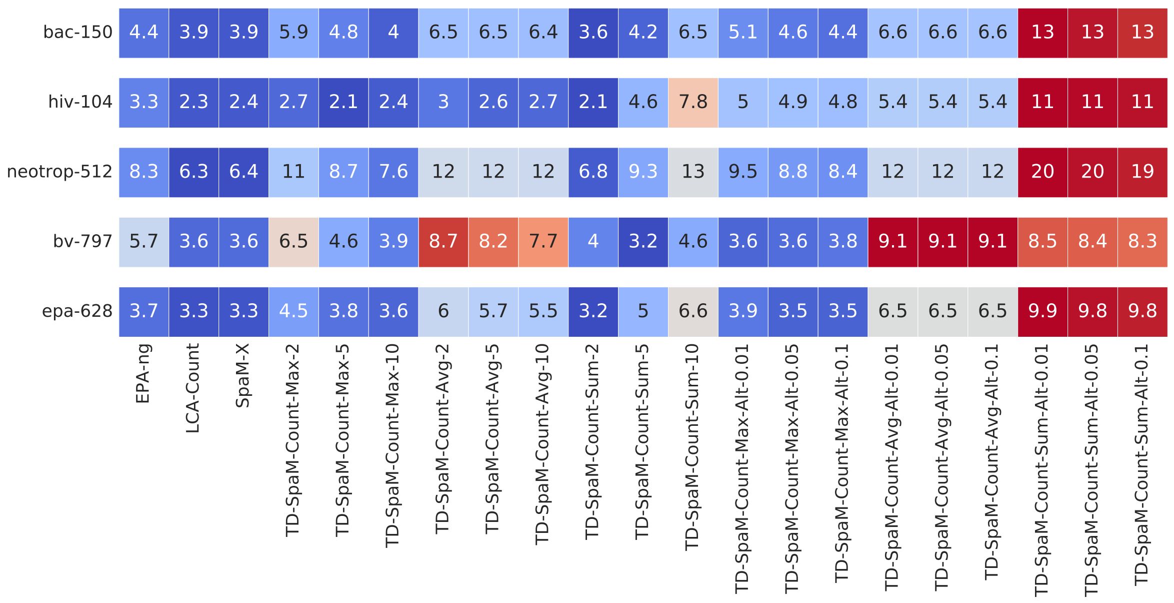


Figure 4.6 – Average accuracy (colored squares) of those heuristics that solely depend on the number of spaced-word matches (x-axis) for five data sets (y-axis). The values for parameter r are appended behind the name of each heuristic. The color coding highlights the performance and is scaled identically to Fig. 4.7.

the most SpaMs resides in a large clade, the average number of SpaMs per reference in this clade remains low, and the search is continued in the 'wrong' subtree instead.

Discussion

Choosing an 'optimal' placement heuristic poses inherent risks: The performance of different placement heuristics varies greatly with different data sets at hand and their associated parameters. The best heuristic for any specific data set is generally unknown and cannot be determined prior to the placement procedure. To select a default heuristic for App-SpaM that performs well in many scenarios, a validation on a broad variety of data sets is necessary. The best performing heuristic of the validation is then chosen as the default. However, the error rate that the selected heuristic achieves on the validation data is an underestimate of its true error rate on unknown data. Thus, it is indispensable to test any default heuristic on unknown data to determine its quality outside the validation environment. In the version of App-SpaM presented in Sec. 3.1 we did not strictly divide the validation and test data. Instead, we evaluated the performance of all heuristics and pattern parameters on bac-150 and hiv-104; both data sets were subsequently also included in the test cases of the overall accuracy evaluation. Hence, the accuracy of App-SpaM on bac-150 and hiv-104 should be taken with caution as the true error may be underestimated. Still, the chosen heuristic proved to provide the best average accuracy across the six remaining data sets as well. Similarly, the default heuristics of several other placement programs besides App-SpaM achieved the best accuracy out of all heuristics in the majority of data sets. This indicates that the selection of the eight data sets is a representative sample of common use cases that were also considered in the evaluation of other programs. As a consequence, we argue that our presented results for all additional placement heuristics are also representative of their performance on unknown metataxonomic data sets.

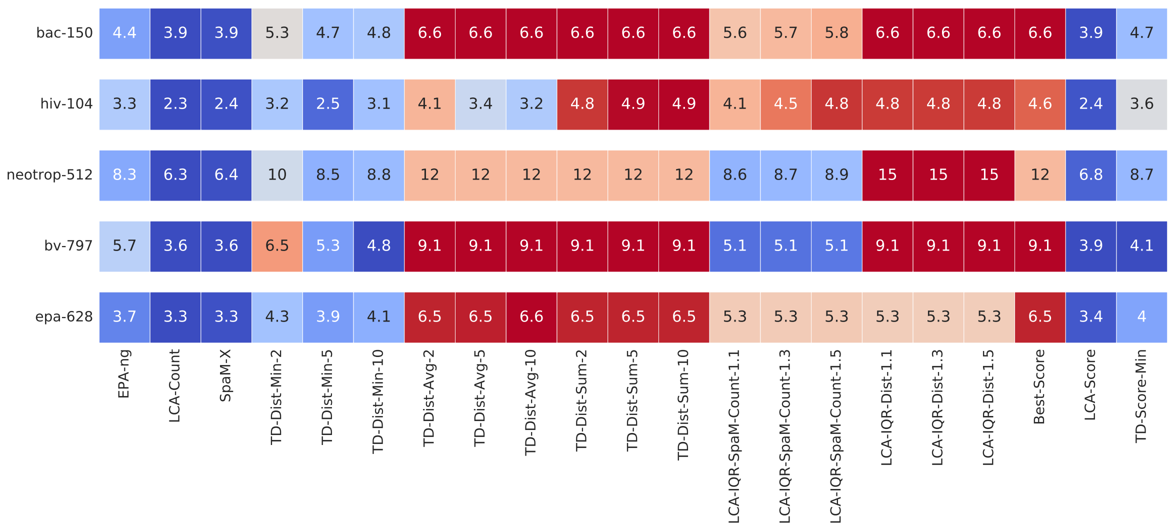


Figure 4.7 – Average accuracy (colored squares) of those heuristics that depend on the distance estimates (x-axis) for five data sets (y-axis). The values for parameter r are appended after the name of each heuristic. The color coding highlights the performance and is scaled identically to Fig. 4.6.

Figure 4.6 and Fig. 4.7 indicate that the placement accuracy of several heuristics varies strongly between different data sets. We suspect that these heuristics exhibit various biases when choosing placement positions: The LCA-approaches will never place a read on a branch directly above a leaf and, vice versa, some heuristics place reads exclusively on branches adjacent to a leaf. Depending on the topology of $\mathcal{T}_{\text{ref}}$, we surmise that such biases affect the placement accuracy positively or negatively, and hence, the accuracy varies for different data sets. Only SpaM-X and the tree descent heuristics overcome this limitation and are able to place query reads anywhere in the tree. Still, they are subject to the topology of $\mathcal{T}_{\text{ref}}$ when s_{avg} or s_{sum} are used to determine the path through which $\mathcal{T}_{\text{ref}}$ is traversed. Thus, we conceptually prefer the tree descent heuristics that use s_{max} or d_{min} as stopping criteria, although they perform worse with respect to the mean ND. All other newly introduced heuristics that utilize distance estimates perform generally worse than those heuristics that rely on the number of spaced-word matches. This result is in accordance with our previous analysis; see Fig. 3.5. The default heuristics LCA-Count and SpaM-X demonstrate a stable performance across all data sets, and we continue to use SpaM-X as default.

4.3 Using Sampling Techniques

SpaM approaches have distinct advantages over other alignment-based methods: They are flexible and versatile with respect to their applications [372] while offering great computational efficiency. They have been demonstrated to outperform other methods with respect to speed and, in several cases, the quality of their results is equivalent or superior [11, 166, 177]. However, there are instances where SpaM approaches cannot compute sequence distances in a reasonable time frame. For example, FSWM reaches its limits on a current standard personal computer when comparing large genomes with multiple gigabases, Read-SpaM is restricted with respect to the coverage of shotgun WGS input data, and large reference data sets comprising thousands of whole genomes cause problems for App-SpaM. An approach to overcome the scalability issue in such cases is the use of *sampling* techniques to reduce the runtime.

The application of sampling methods for alignment-free programs is a common idea to limit the amount of data that must be processed. Especially for word-based methods, the sampling of k-mers greatly reduces the time and space complexity. It has been suggested that a k-mer subset of appropriate size is often sufficient to estimate the similarity between sequences to a satisfactory degree: For example, MASH [149] estimates the distance between two sequences by calculating the Jaccard index of their k-mer sets, as discussed in Subsec. 2.4.2. Subsequently, the Jaccard index is used to infer the Hamming distance. However, computing the Jaccard index based on the complete sequence data may be difficult, as the total number of k-mers is large for long sequences. Thus, instead of comparing the complete k-mer sets, MASH creates so-called *sketches* of the sequences by taking random samples from their k-mer sets. The Jaccard index is then calculated directly from these sketches; the result is an unbiased estimate for the real Jaccard index. The expected error of the estimator depends only on the size of the sketches and not on the size of the original sequences [373], making the approach suitable for whole genome comparisons. The resulting distance matrices are used, for example, to construct phylogenetic trees [374].

In general, hashing techniques that summarize sequences by means of sketches are termed *locality-sensitive hashing* (LSH) methods [375]. These techniques achieve sublinear space requirements and execution times with respect to the sequence length. For LSH techniques, the similarity of the original sequences is estimated from the similarity of the created sketches. LSH techniques have been used with great success, for example, to pre-filter reference databases for the most promising reference sequences in metagenomic profiling [376], to screen sequencing reads for their taxonomic diversity [377], or to scan sequences for the most promising alignment sites under the presence of uneven k-mer distributions [378].

Given the extensive existing research on sampling techniques for word-based sequence similarity methods, it is evident that sampling could also be a tool to overcome the limitations of SpaM approaches by reducing the amount of data that is processed. Although SpaM approaches demonstrate reasonable scalability, especially compared to alignment-based methods, they are impeded by the quadratically growing number of background matches for long input sequences. Additionally, it is unclear how many homologous spaced words are required to robustly estimate distances between sequences. It may well be that the distance estimates do not improve after a certain number of homologous spaced-word matches have been found. However, the number of homologous spaced words grows with increasing sequence similarity and the length of homologous sequence segments.

In general, sampling is performed either among spaced words or among filtered (or unfiltered) spaced-word matches. Here, with sampling, we refer to the process of selecting a subset of all spaced words, or of all spaced-word matches, respectively. For a given sequence S of length n and for the underlying pattern P of length l, the total number of spaced-word occurrences extracted from S for further processing is $n - l + 1$: one spaced word starts at every position of S. The number of spaced-word occurrences, thus, is merely determined by the length of S. We refer to the set of all spaced-word occurrences of S as $\mathcal{S}$, hence $|\mathcal{S}| = n - l + 1$. A sampling procedure $\mathcal{R}$ chooses a subset $\mathcal{S}' \subset \mathcal{S}$ from the occurrences of S. We call $r = \frac{|\mathcal{S}'|}{|\mathcal{S}|}$ the *sampling ratio* of $\mathcal{R}$. Inherently, sampling among $\mathcal{S}$ also reduces the number of spaced-word matches that are generated. Depending on the implementation of the underlying SpaM approach and on the type of sampling strategy that $\mathcal{R}$ uses, the gain in computational time may be particularly large: Depending on the implementation, evaluating spaced-word matches may require consecutive memory access from different regions in the main memory. This kind of memory access poses a bottleneck in SpaM approaches because the underlying computational system can perform hardly any cache optimization.

One simple method to sample spaced words is to deliberately omit some spaced words for a sequence S directly when reading the sequence. To reach a given sampling ratio r, we consider two simple methods: Either, only those spaced words that occur at a position of S that is a multiple of $\frac{1}{r}$ are approved and all others are discarded; we term this approach the *position-based* sampling ($\mathcal{R}_{\text{pos}}$). Or, each potential spaced word is approved with a probability of r, and discarded otherwise; we term this approach the *randomized* sampling ($\mathcal{R}_{\text{rand}}$). The difference of the two approaches lies in the distribution of the positions of the retained spaced words in $\mathcal{S}'$ across S: For position-based sampling, the chosen words are equally spaced along the given sequence and no computational overhead is necessary for the selection process itself. On the contrary, the positions are chosen at random in $\mathcal{R}_{\text{rand}}$ and, thus, the retained spaced words represent a random sample with a uniform distribution from $\mathcal{S}$. For both approaches, however, there are unfavourable implications on the number of resulting spaced-word matches: When sampling spaced words from two input sequences S_1, S_2 with $\mathcal{R}_{\text{pos}}$ and a sampling ratio of r, the amount of spaced-word matches depends heavily on the number and location of indels, as well as their starting points. Assuming two identical sequences, all spaced words chosen by $\mathcal{R}_{\text{pos}}$ are homologous and form spaced-word matches. However, in any real-world scenario, it is likely that the chosen spaced-word matches do not correspond to homologous positions due to indels or different start sites of the two sequences. Thus, we assume that the number of spaced-word matches is drastically reduced for $\mathcal{R}_{\text{pos}}$. For $\mathcal{R}_{\text{rand}}$ with a sampling ratio r, the amount of homologous spaced-word matches drops by a factor of r^2. The reason is that a spaced word in S_1 with a homologous position in S_2 is chosen in both sequences only with a probability of $\frac{1}{r}$. Hence, the decrease in the number of spaced-word matches is detrimental in both methods, as the amount of lost information increases disproportionately to the gain in computational complexity with respect to speed and memory.

Hence, it is essential to use a sampling method that overcomes this limitation. A suitable approach for this is hash-based sampling [375]. The idea is to choose the *same* subset of spaced words with respect to their match positions from both sequences to ensure that potential homologous SpaMs are retained. Consequently, each SpaM that is present when all spaced words are used is still present when the according spaced word is sampled. More specifically, the nucleotides at the match positions of a spaced-word occurrence are represented as a 32 bit integer; subsequently, this is referred to as its *key* K. Given a hash function h and applying it to K, its *value* is $h(K) \in [0, h_{\max}]$. We define the sampling method $\mathcal{R}_{\text{hash}}$ with two parameters z and t as follows: A spaced word—represented by K—is considered as part of the sample if $h(K \oplus z) < t$. Otherwise, the spaced word is discarded. Here, $\oplus$ is the bit-wise XOR-operation: the match positions are scrambled with a fixed but randomly chosen 32 bit integer z. This ensures that the resulting values of h are not dependent on the nucleotide composition of S. The parameter t is the *sampling threshold* that adjusts how many spaced words are retained. If the values of h are assumed to be uniformly distributed between 0 and $h_{\max}$ and, furthermore, it is assumed that all spaced words occur on average with identical frequency, a desired sampling ratio r is achieved by setting the threshold t as $r \cdot h_{\max}$. In practice, we use a fast implementation of the CRC32 algorithm [379] as h. Thus, the first assumption is met because the values of $h(K \oplus z)$ are uniformly distributed. The second assumption is violated in most real-world cases, since spaced words do not occur with equal frequencies. However, a slight deviation in the obtained sampling ratio does not influence our proposed procedure in practice, see below.

Results

We evaluated the use of the hash-based sampling technique $\mathcal{R}_{\text{hash}}$ within App-SpaM. For this, we compared the accuracy of App-SpaM's placement with and without applying $\mathcal{R}_{\text{hash}}$ on three different data sets. We used varying values for the sampling ratio r that determines the sampling threshold t. Thereby, we were able to study the loss of accuracy depending on r with respect to different data sets. Here, we only investigate how the quality of the *placements* changes when sampling is applied. We do not evaluate the accuracy of the derived distance estimates; this was already done in a similar study for the use of sampling techniques in the software FSWM [370].

Figure 4.8 shows the loss of accuracy of App-SpaM on the vir-104 data set. We varied r from 0.1 (on average, 10% of all spaced-words are used) to 1 (all spaced words are used) in steps of 0.1. For each value of r we recorded the average accuracy of App-SpaM's placements over the same 50 random prunings. The accuracy is measured by means of the node distance. When all spaced words are taken into account, the accuracy of App-SpaM is 3.18. The accuracy slowly declines with lower sampling ratios and reaches its minimum at $r = 0.1$ with a node distance of 4.05. Large standard deviations arise from the wide variation of scenarios that occur in the random pruning process of the PEWO software. Standard deviations of similar extent were measured for all placement programs in the PEWO evaluation procedure, see Subsec. 3.1.2. As identical prunings were used for all sampling ratios, the increase in standard deviation for lower sampling ratios is explained by the sampling itself. The number of spaced-word matches decreases linearly with decreasing sampling ratio, as expected.

Analogous to Fig. 4.8, Fig. 4.9 displays the accuracy of App-SpaM for sampling ratios from 0.0001 to 1 in logarithmic steps. For small sampling ratios (1% and below), the accuracy of App-SpaM declines strongly. For sampling ratios of 0.1% and below, the quality of placements is as poor as that of the root control method, where each query is simply placed at the root (compare Fig. 3.12). Furthermore, the standard deviation increases strongly for decreasing values of r. Both these observations are explained by the number of spaced-word matches that arise for low sampling ratios: Although there are approximately 200 matches

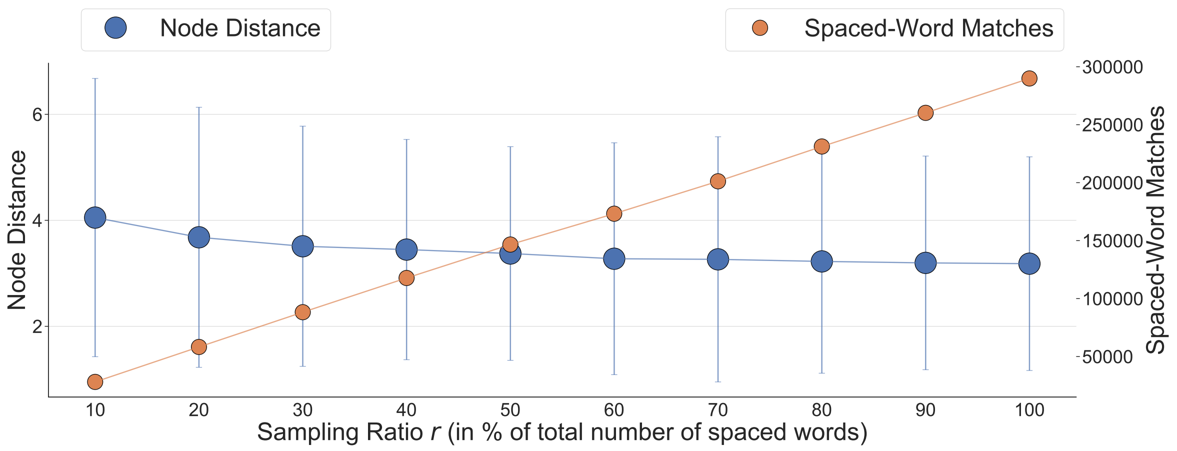


Figure 4.8 – Accuracy of App-SpaM depending on the sampling ratio r on the vir-104 data set. The sampling ratio is given as the percentage of spaced words that were used (x-axis). The average node distance across 50 random prunings (left y-axis, blue) and the number of spaced-word matches (right y-axis, orange) are given. Vertical blue lines indicate the standard deviation of a given value of r across 50 prunings.

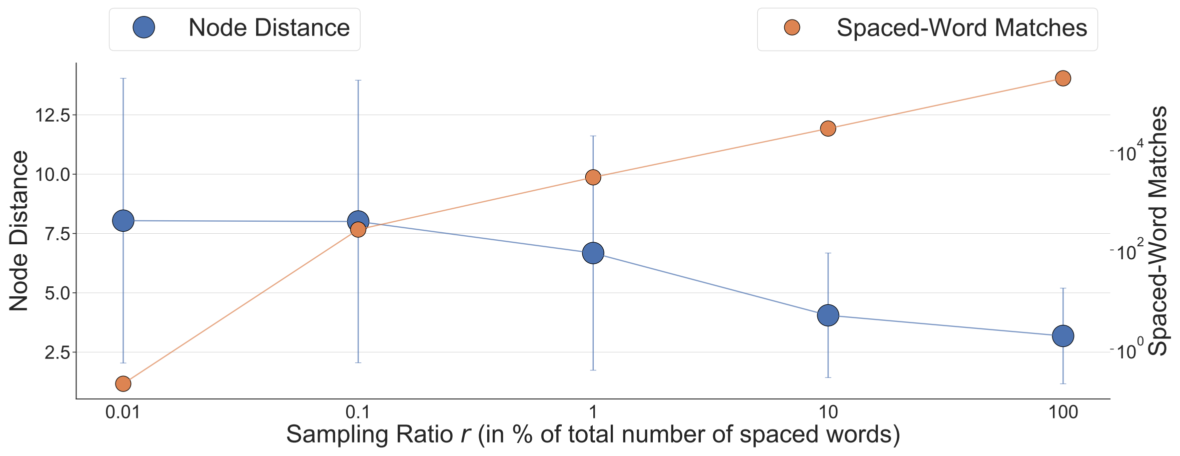
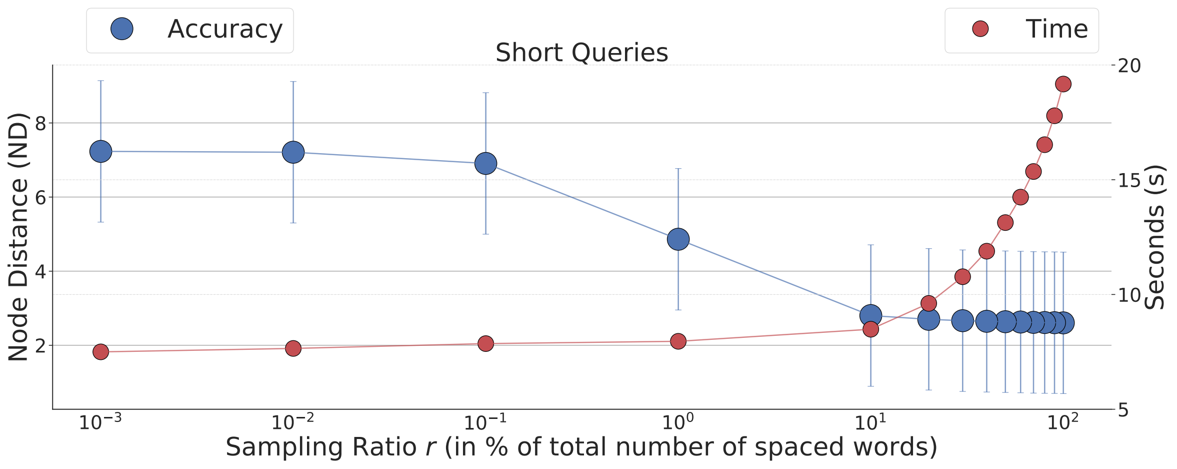


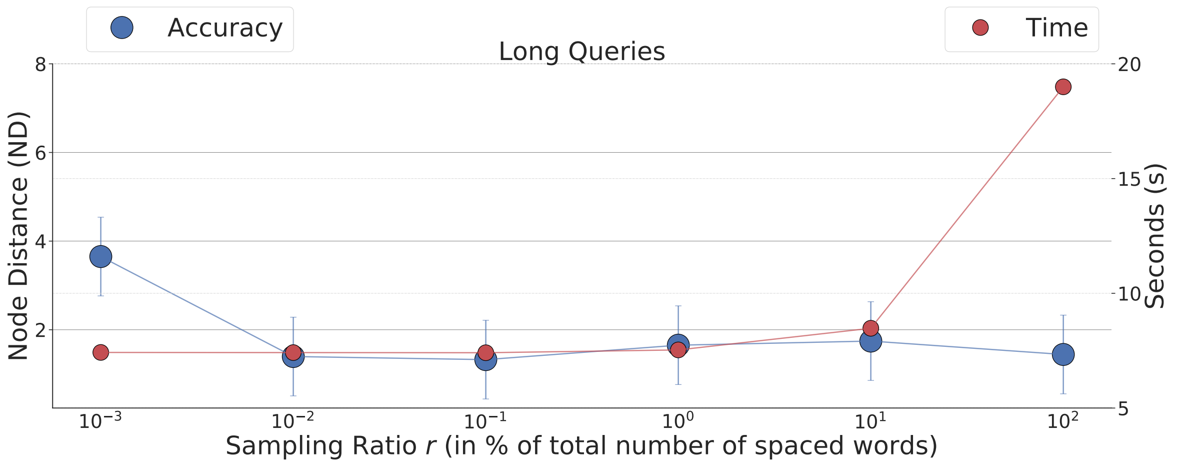
Figure 4.9 – Accuracy of App-SpaM depending on the sampling ratio r on the vir-104 data set. The sampling ratio is given as the percentage of spaced words that were used. The average node distance across 50 random prunings (left y-axis, blue) and the number of spaced-word matches (right y-axis (log-scale), orange) are shown. Vertical blue lines indicate the standard deviation for a given r across the 50 prunings.

for $r = 0.001$ across all reads to be placed, for $r = 0.0001$ there is less than one match found per pruning for all to-be-placed reads. Therefore, there is no remaining information to infer proper placement positions when low sampling ratios are used for metataxonomic data sets. The accuracy of App-SpaM remains comparable between different placement modes and weights when $\mathcal{R}_{\text{hash}}$ is applied, as shown in Suppl. Fig. B.14. Similarly, the trends that we observed on the hiv-104 data set also hold for the bac-150 and tara-3748 data sets; see Suppl. Fig. B.15.

To conclude the above experiments, we also ran App-SpaM on the wol-43 data set that comprises whole prokaryotic genomes. We performed a leave-one-out evaluation for each of the 43 reference genomes and evaluated two different scenarios: First, we partitioned the pruned reference sequence in each pruning experiment into short reads with 150 bp lengths and placed each read individually. Second, we placed the singular pruned reference without further processing onto $\mathcal{T}_{\text{ref}}$. Figure 4.10 illustrates the accuracy of App-SpaM across sampling ratios from 0.01% up to 100% as well as runtimes for both experimental setups. When placing short reads, the results are in accordance with those from previous data sets, see Fig. 4.10a. The accuracy of App-SpaM remains near constant for a sampling ratio between $r = 1$ and $r = 0.1$; for lower values of r, the accuracy drops substantially. The largest gain in computation speed occurs for large sampling ratios: The runtime drops by 56% from $r = 1$ to $r = 0.1$; in comparison, it drops only by 12% from $r = 0.1$ to $r = 0.00001$. When single whole genomes are used as query sequences instead of multiple short reads, the results deviate substantially from previous observations, as shown in Fig. 4.10b. Overall accuracy improves significantly for all values of r and remains steady even for low sampling ratios. For $r = 1$ the ND is 1.8 times higher when the complete genome is divided into short reads. There is no degradation of the accuracy up until $r = 0.0001$. In contrast, the gain in computation speed is identical to what was observed previously.



(a) The pruned genomes are partitioned into short reads with length 150 bp and each read is placed individually.



(b) Here, the pruned genome is placed as a single query sequence onto $\mathcal{T}_{\text{ref}}$.

Figure 4.10 – Accuracy of App-SpaM depending on the sampling ratio r on the wol-43 data set. The sampling ratio is given as the percentage of spaced words that were used. The average node distance across a leave-one-out evaluation (left y-axis, blue) and the runtime (right y-axis (log-scale), red) are shown. Vertical blue lines indicate the standard deviation across all experiments. The upper part (a) shows results for short query sequences of 150 bp while the lower part (b) shows results for query sequences that span the entire pruned genome.

Discussion

A given sampling strategy has to be evaluated primarily with respect to the change in accuracy and speed in contrast to not using any sampling. The loss in accuracy describes the degree to which distance estimates deteriorate in dependence on the sampling ratio or total size of $\mathcal{S}'$. Here, another important attribute to consider is the robustness of the distance estimates, i.e., the standard deviation across multiple repeats with different sampled subsets. The speed can be measured as the absolute amount of time required for the computation, or as the speed up that could be gained over the according SpaM approach without sampling. In the optimal case, the accuracy of the results is not directly dependent on r, but instead only on $|\mathcal{S}'|$, implying that a constant number of randomly chosen spaced words is sufficient to determine sequence similarity to a satisfactory degree. In this case, the computational complexity of the algorithm is no longer dependent on the length of the input sequences, but instead only on the number of input sequences.

We applied the min-hash sampling technique to App-SpaM to reduce the number of spaced-word matches that need to be processed, and accordingly, its runtime. By using the min-hash principle, the number of SpaMs is reduced linearly with the predefined sampling ratio r. This approach is common among word-based methods for metagenomic tasks when large quantities of sequence data are processed. Although it was already examined how various sampling techniques influence the distance estimates of the software FSWM [370], the applications of FSWM and App-SpaM are fundamentally different. For FSWM the input sequences comprise complete genomes; this allows the use of low sampling ratios without losing essential information. In such a scenario, the use of sampling is comparable to a *sketching* approach, where a sequence is summarized by a compact representation (the set $\mathcal{S}'$ of sampled spaced words). Whereas sampling in FSWM had great success, the results presented here highlight that sampling is of little benefit when App-SpaM is applied to short and independent sequencing reads from metagenomic experiments. The number of spaced-word matches per read drops rapidly, and consequently, it becomes impossible to determine precise placement locations for each short read. However, when App-SpaM is applied for long queries (for example, whole genes, draft genomes, or scaffolds), sampling is an attractive opportunity to handle large amounts of query sequences because low sampling ratios r do not entail a low number of spaced-word matches.

Thus, the usefulness of min-hash sampling in App-SpaM depends on the use case at hand: The benefit of sampling techniques is limited when short query sequences are present. Although the speedup is linear with respect to the sampling ratio r, the accuracy is constrained by the average number of spaced-word matches obtained per query sequence. However, the number of SpaMs also declines linearly with the sampling ratio r. To grasp the influence of this effect when short query sequences are present, it is useful to consider a simplified example that highlights the decline of expected SpaMs: For a sampling ratio of $r = 0.1$ and a query sequence S_q with 150 bp, the number of spaced-word matches between the query and an identical reference drops to only 10.7 matches on average. Lowering the sampling ratio to $r = 0.01$ results in only 1.7 expected matches per query. Furthermore, it is unlikely that the most closely related reference sequence represents exactly the organism from which S_q originates. Instead, the true number of SpaMs is presumably lower than the stated numbers. Given these considerations, it seems unreasonable to expect App-SpaM to infer proper placement positions of short query reads when sampling is applied. Only when the length of the query sequences increases, sampling becomes worthwhile.

The presented results are in accordance with this proposition: First, the number of spaced-word matches exhibits a linear decrease with respect to the sampling ratio r as implied by Fig. 4.8 and Fig. 4.9. Second, the accuracy of App-SpaM drops significantly for low sampling ratios $r < 0.1$ when query reads are short. This is the norm when dealing with metataxonomic data and implies that sampling is of limited use in this area of application. However, App-SpaM does not require sampling in this use case anyway: When query and reference sequences comprise single marker genes, App-SpaM is the fastest program to perform phylogenetic placement and its speed is *not* limited by the amount of SpaMs. In contrast, sampling is of great use when reference and query sequences comprise long genomes as visualized in Fig. 4.10. Here, the accuracy of App-SpaM remains stable up to a sampling ratio of $r = 0.0001$. In Fig. 4.10b, the largest time gain is between $r = 1$ and $r = 0.1$, and the time remains close to constant for all ratios $r \leq 0.1$. This is due to the fact that the runtime of App-SpaM is dominated by processing steps other than the comparison of SpaMs when only a single query sequence is present. When using multiple query sequences, the runtime exhibits a reduction across the entire sampling range, see Suppl. Fig. B.16. This makes sampling with low ratios r extremely powerful for data sets of ever-increasing sizes.

Sampling of SpaMs is another piece of the puzzle that enhances App-SpaM, with the caveat that it is only useful when query sequences have a sufficient length. This prerequisite is commonly met, for example, when bags of reads, draft genomes, or scaffolds are placed onto reference phylogenies. The proposed sampling technique is available in the newest publicly available version of App-SpaM and may be applied to appropriate data sets. Although it would be best to enforce a minimal number of matches for each read that must be obtained when sampling is applied, the current implementation simply uses the supplied sampling ratio r. Thus, end users have to be aware of the implied changes in accuracy, depending on the type of input data used.

4.4 Assessing Placement Uncertainty

Instead of indicating a *single* placement position for a query S_q, placement programs may provide *multiple* potential placement locations for S_q and weight them according to their plausibility or *uncertainty.* For maximum likelihood-based placement programs, this approach seems evident as they already calculate likelihood values for several or all branches of the reference tree. Then, they simply infer a weight for each placement from the likelihood value associated with it: The weight of each placement location is its likelihood normalized by the total sum of likelihoods across all placement locations; by this, the sum of placement weights for a query sums up to 1. Accordingly, the weight is also referred to as *likelihood-weight ratio* (LWR). The weights for a query can be interpreted as a probability distribution across the branches of $\mathcal{T}_{\text{ref}}$. As such, they also provide an opportunity to assess the uncertainty concerning the placement of S_q: If multiple placement locations with similar weights are specified for S_q, its correct placement position is uncertain. Vice versa, if most of the weight is assigned to a single location, the according query placement is more certain. LWR values have been demonstrated to correlate strongly with posterior probabilities and also with the precision of placements [12]. The uncertainty of placements may guide decision processes in subsequent analyses; for example, it might be sensible to discard queries with uncertain placements from further analysis steps, such as the estimation of metagenomic sample composition or taxonomic identification.

The declaration of multiple weighted placement positions for a single query is not limited to ML-based programs in principle. For example, the phylo-k-mer-based approach RAPPAS also generates multiple placement positions with LWR-like weights for each query. For this, RAPPAS calculates a measure of placement uncertainty from the weighted distribution of the ancestral k-mer sequences at inner nodes of $\mathcal{T}_{\text{ref}}$ that are associated with the query. In contrast, both methods APPLES and App-SpaM do not inherently provide the possibility to specify placement uncertainty. However, App-SpaM saves the number of spaced-word matches as well as the pairwise distance estimates between S_q and each reference sequence. We consider these two sources of information as *leaf annotations* in $\mathcal{T}_{\text{ref}}$ as each leaf represents a single reference sequence. Just like for placement heuristics, the leaf annotations then provide information that may serve as a basis for inferring LWR-like values. The uncertainty of a placement may be specified as a function of the leaf annotations in the subtree below the placement branch, or of all leaf annotations in the whole tree. For example, such a function may incorporate the difference in the number of SpaMs between the two subtrees below the placement branch, or the relative number of SpaMs below the placement branch with respect to the total number of observed SpaMs across all reference sequences.

Methods

We extended App-SpaM with the ability to specify multiple placement locations for a single query that are weighted according to their certainty. In accordance with RAPPAS and the standardized JPlace file format, we refer to these weights as likelihood-weight ratios (LWRs) even though they are not derived from likelihoods. This ensures that several subsequent programs interpret these LWR-like weights as a probability distribution of potential placement locations over the branches of $\mathcal{T}_{\text{ref}}$. To infer the weights, we propose two weighting schemes for the inner nodes of $\mathcal{T}_{\text{ref}}$ that are based on the leaf annotations. Each weighting scheme simultaneously constitutes a new placement heuristic in addition to those presented in Sec. 4.2. The underlying process for all new heuristics is identical and analogous to ML-based programs: A query is placed on every branch of $\mathcal{T}_{\text{ref}}$ resulting in $2 \cdot (m-3)$ placement positions for a tree with m organisms. Subsequently, the branch b of each placement position receives a weight w_b that indicates the plausibility that S_q really belongs onto branch b. Then the weights of all branches are normalized, resulting in a probability distribution across all branches in $\mathcal{T}_{\text{ref}}$. The best five placement positions are reported in the JPlace output file; the number of reported positions can be modified when running App-SpaM with these modes.

Similar to the placement heuristics introduced earlier, each weight w_b for a branch b is dependent on the distribution of the SpaMs below b. We refer to the subtree that is induced by b as T_b. Let u and v be the two distal branches directly below b; we define the set $\mathcal{S}_b = \{S_i \mid S_i \in T_b\}$ to comprise all reference sequences below b. We refer to the total number of SpaMs that occurred between S_q and any reference in T_b as

$$s_b = \sum_{S_i \in \mathcal{S}_b} s(S_i, S_q)\,. \tag{4.11}$$

The first heuristic assumes that a placement on b is uncertain when the SpaMs below b are equally distributed among the two subtrees T_u and T_v. On the other hand, if the SpaMs of most references in T_b belong to either T_u or T_v, the placement is rather certain. This measure of *dispersion* is then multiplied by the relative number of SpaMs that occurred in T_b:

$$w_b = \underbrace{\left(1 - \frac{|s_u - s_v|}{s_u + s_v}\right)}_{\text{dispersion}} \cdot \left(\frac{s_b}{s_{\text{root}}}\right)\,. \tag{4.12}$$

Thus, the weight of a branch located directly above a leaf is given as the relative number of SpaMs of the respective reference sequence.

For inner branches, the weight approaches zero if all SpaMs below b belong to the same subtree; in this case, a better placement is expected within the subtree where the SpaMs occur. In contrast, if the SpaMs are equally distributed among the subtrees, the weight evaluates to the relative weight of all SpaMs in T_b. The second heuristic uses the minimal number of SpaMs in T_b to determine the placement certainty of S_q weighted by $|\mathcal{S}_b|$, the total number of leaves in T_b. Here, the assumption is that the minimal number of SpaMs to any reference sequence influences the quality of placement results, as discussed in Subsec. 3.1.3. Simultaneously, the probability of a reference with a low number of SpaMs increases with the size of T_b:

$$w_b = \min_{S_i \in \mathcal{S}_b} s(S_i, S_q) \cdot |\mathcal{S}_b|\,. \tag{4.13}$$

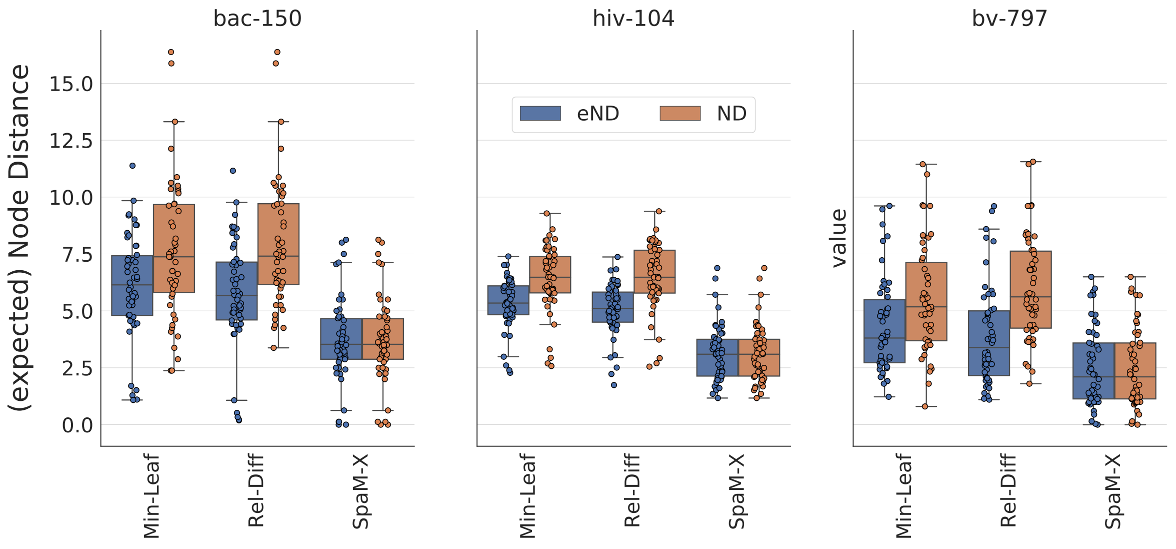


Figure 4.11 – Expected node distance (blue) compared to the node distance (orange) of App-SpaM on three different data sets (three subplots) for three different heuristics (x-axes). Accuracy was measured using the pruning-based accuracy evaluation with 50 repetitions (each dot is the average (expected) node distance of a single repetition).

We refer to the first placement heuristic in Eq. 4.12 as Rel-Diff and to the second heuristic defined by Eq. 4.13 as Min-Leaf.

Discussion

The evaluation of LWR-like values is challenging because each query read has a single clearly defined position in $\mathcal{T}_{\text{ref}}$. For this purpose, the PEWO framework provides the *expected ND* (eND) metric. The eND is defined as the sum of NDs across all proposed placements for S_q weighted by their LWRs. If the LWRs are of high quality, the eND distance is expected to outperform the ND metric over multiple pruning experiments: The average distance across the weighted placement positions is a better estimate of the true placement position than a single placement position is on average. Correspondingly, ML-based programs exhibit an improved accuracy under the eND metric compared to the ND metric, see our comprehensive evaluation in Sec. 3.1.2 and Suppl. Fig. A.2 to Suppl. Fig. A.9. When subsequent processing steps require a singular placement branch, only the placement with the highest LWR is used. We evaluated Rel-Diff and Min-Leaf by running the PAC workflow on the bac-150, hiv-104, and bv-797 data sets with 50 iterations each. Besides the two new heuristics, placements were also carried out with the default version of App-SpaM.

Figure 4.11 shows the accuracy of our two proposed methods with respect to SpaM-X over 50 pruning iterations on each of the three data sets. Both new heuristics perform significantly worse than App-SpaM's default version with node distances that are at least twice as high on all data sets. For both new heuristic, eND values are significantly better than ND values, indicating that the weighted average placements are superior to the single placement with highest weight. These unsatisfactory results demonstrate that the suggested placement heuristics do not work as intended. Although both methods outperform the control methods introduced in Sec. 3.1.1, the improvement remains small. For example, averaged over all queries, the highest weighted placement by Rel-Diff on the hiv-104 data set is only

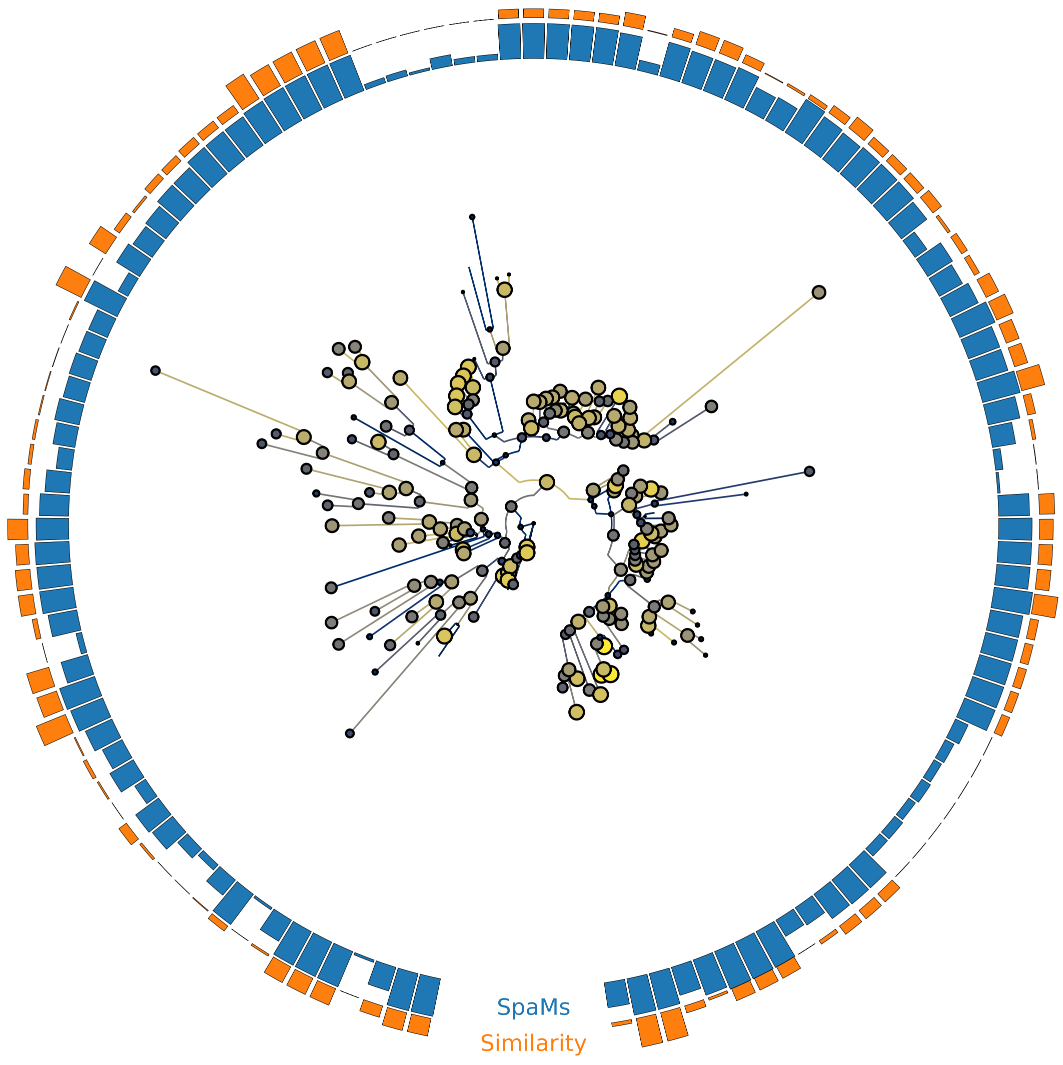


Figure 4.12 – Tree with LWR-like values on the BAC-150 data set. APP-SPAM infers multiple weighted placement positions for a single query S_q on the BAC-150 data set. The two outer rings highlight the leaf annotations from APP-SPAM's internal algorithm: The bars of the inner ring (blue) indicate the number of spaced word matches between S_q to each reference. The height of each bar is normalized by the maximal number of SpaMs that occur to any reference sequence. The bars of the outer ring (orange) indicate the similarity of S_q to each reference. The size and color of each node indicates the weight of the branch above.

0.9 nodes better than placing each query at the midpoint of the tree. With respect to the expected node distance, weighted placements of REL-DIFF improve on average by 2.4 nodes compared to the midpoint control method. Similar effect sizes are prevalent for BAC-150 and BV-797.

One reason for the loss in accuracy is visualized in Fig. 4.12: Often, the number of SpaMs are only marginally different between multiple reference sequences distributed among the whole reference tree. In those cases, the proposed weighting schemes also infer similar

LWR-like values for nodes across the whole reference tree instead of local groups of nodes. The figure illustrates such an example for a single query S_q and a single pruning experiment on the BAC-150 data set. One counter example where placement accuracy is high is given in Suppl. Fig. B.17 for the BV-797 data set; here, the leaf annotations create little ambiguity about fitting placement positions. The difference to the example in Fig. 4.12 is that reference sequences are evolutionarily less related in BV-797. In those cases where a query sequence produces a high number of SpaMs to multiple or all references, it becomes all the more important to determine those references that indeed have the highest number of SpaMs, as done by the SPAM-X heuristic. Another approach could be to include the distance estimates in addition to number of SpaMs. As visualized in Fig. 4.12, inferred distances exhibit a larger variation across the reference sequences. However, we did not devise or test any joint measures that produce adequate LWR values and perform better than the two proposed heuristics so far. In this context, Suppl. Fig. B.17 also highlights another problem: In certain cases, a low number of *highly similar* SpaMs are reported between the query and a distant reference sequence. The discrepancy between the number of SpaMs and the corresponding distance estimates renders the combination of both difficult. Our current approach to infer LWRs from the distribution of leaf annotations might be fundamentally flawed. After all, the leaf annotations might provide too little information to infer reasonable weights for inner nodes of the reference tree. Besides combining SpaM counts and distance estimates, two additional approaches may be pursued in future research.

First, deriving additional information about inner nodes could be achieved by employing *phylo-k-mers*. The software RAPPAS first introduced phylo-k-mers for the task of phylogenetic placement. In RAPPAS, a phylo-k-mer is a word of length k that is derived from the reference sequences and carries phylogenetic information which can be utilized for placement, see Subsec. 2.7.1. We adapted this idea to the application of *spaced* phylo-k-mers and evaluated whether spaced phylo-k-mers can guide the placement process [380]. To account for potential ancient k-mers that are not present in the given references, RAPPAS infers a set of appropriate phylo-k-mers via ancestral sequence reconstruction. When using spaced phylo-k-mers, it is instead acceptable to use a sufficiently low pattern weight w. More specifically, we annotate each inner branch b of $\mathcal{T}_{\text{ref}}$ with spaced words that occur exclusively in the subtree T_b. These monophyletically occurring spaced words characterize the references in the respective clades. After computing a database of spaced phylo-k-mers in a preprocessing step, they guide the placement process. While the integration of spaced phylo-k-mers into the current version of APP-SPAM is challenging, it might alleviate the troubles that we experienced with the heuristics REL-DIFF and MIN-LEAF.

Second, placement uncertainty may be quantified for existing heuristics, such as SPAM-X, by using adapted bootstrapping approaches. Recently, it has been demonstrated that non-parametric bootstrapping works well for alignment-free PP programs, especially for long query sequences [381]. Similarly, randomly selected subsets of spaced words across multiple program runs can be used to assess the variation of placement positions. The variation then constitutes a measure for the uncertainty associated with the resulting placements. Outside of metataxonomics, this idea is similar to placing multiple genes of the same species to infer an overall species position, see Chpt. 5. In summary, the indication of placement uncertainty in alignment-free placement programs is not as evident as for Maximum likelihood (ML)-based programs. Overall, our experiments revealed a strong drop in accuracy for the two newly introduced heuristics that quantify placement uncertainty and they are not fit for use in their current state.

Chapter 5

Iterative Update of Phylogenetic Trees

Matthias Blanke conceived, planned, and supervised all work presented in this chapter. The implementation and execution of experiments in Sec. 5.1 was carried out jointly by Dark Engel (Department of Bioinformatics, Georg-August-Universität Göttingen) and Matthias Blanke. The implementation and execution of experiments in Sec. 5.2 was carried out jointly by Patricia-Franziska Römer (Department of Bioinformatics, Georg-August-Universität Göttingen) and Matthias Blanke.

The characterization of short sequencing reads from metagenomic amplicon experiments has been the main focus of phylogenetic placement algorithms [10, 12, 15, 335], of associated software that deals with placement data [194, 334, 337, 339, 382], and of studies that use phylogenetic placement software [383, 384]. This confined use is caused by the high computational demands of existing placement programs and their dependence on sequence alignments, which limits their applicability and excludes sequencing data that span longer regions, or are only available as short reads or incomplete scaffolds; such sequencing data are commonly generated, for example, from metagenome assembled genomes [9], or from low to ultra-low sequencing experiments in whole genome association studies or conservation biology [8, 385]. Studies with the purpose of finding phylogenetic relationships for a single species often apply de-novo tree reconstruction methods based on multiple marker genes [386–390]. Although these studies also term their process as 'phylogenetic placement', they do not perform rigorous placement into existing trees as described in Sec. 2.7. Consequently, they also do not use available high-quality trees and the respective software tools tailored for phylogenetic placement. Although de-novo reconstruction can be more accurate than phylogenetic placement, de-novo reconstruction involves many resource-intensive steps that are in a disproportionate relation to the potential gain in accuracy. These steps involve the collection, processing, and curation of sequencing data, as well as the computation of the reference tree. Each of these steps requires careful consideration to mitigate errors that may distort the resulting tree. On the contrary, the use of existing trees, which were calculated from curated data sets and built by experts from the respective research communities, illustrates the potential benefits of placement. Performing real phylogenetic placement in these instances also facilitates the efficient screening of large amounts of new sequencing data that cannot be processed with more time-intensive methods.

The modest usage of phylogenetic placement in a phylogenomic context may also be explained by the fact that PP programs were not designed to handle unstructured sequence data up until now. However, an algorithm such as App-SpaM enables the augmentation of existing phylogenetic species trees with a single or a whole set of species based on arbitrary sequence data, such as their entire genome, a set of genes, or a collection of short reads. This novel opportunity allows the application of phylogenetic placement outside the metagenomic field. An obvious usage of App-SpaM is the continuous augmentation of phylogenetic trees with new species, but there are various other emerging uses for phylogenetic placement where reference sequences may be unaligned. In the following sections, we present use cases in which App-SpaM has distinct advantages over ML-based programs and evaluate its accuracy and feasibility for these tasks.

5.1 Continuous Augmentation of Phylogenetic Trees

The field of phylogenetics addresses the reconstruction of phylogenetic relationships based on molecular data from a set of species, usually by reconstructing a phylogenetic tree [391] or a phylogenetic network [392]. When sequencing data was still sparse, the phylogenies of species were reconstructed on the basis of single genes, called *marker genes*, which were determined to provide a sufficient resolution to identify the relationships between all species under investigation. However, phylogenetic trees that are based on only a single gene are not necessarily representative of the evolutionary relationships of the species itself. The history of a single gene deviates, sometimes profoundly, from the history of the species itself. Furthermore, it is difficult to choose the appropriate gene for phylogenetic analyses, and sensible genes vary between different groups of species [393–395]. Finally, another inherent problem is the limited resolution that a single gene offers: The number of available sequence positions is scarce for a single marker gene, which may cause statistical errors in the phylogenetic estimation process, especially if a large number of species is considered in the analysis. To overcome these limitations and as more sequencing data became readily available, the approach has shifted to the use of multiple genes or even the complete genomic data from the species at hand, to determine their phylogenetic relationship. This field is also referred to as *phylogenomics* and it is commonly accepted that increasing the amount of genomic data results in phylogenetic trees that are better resolved and highly supported [396–398] and there have been many theoretical and practical approaches to reconstruct phylogenies based on whole genome sequencing data [397–399]. However, the use of increasing amounts of data may also result in systematic biases, for example, due to an increasing level of noise or due to model violation [400]. Thus, it is not always possible or feasible to use the complete genomic data of all species [401]. Therefore, it is common practice to use a large number of carefully selected homologous genes from all organisms of interest to infer a phylogenetic tree [402]. There are two primary approaches to using gene sequences for the reconstruction of species phylogenies: One straightforward approach is to construct an alignment for every gene and subsequently concatenate all gene alignments into one single *super-alignment*. These approaches have also been termed *supermatrix* methods [403]. Methods for character-based phylogenetic reconstruction, such as maximum likelihood or maximum parsimony, are then used on the super-alignment as presented earlier. There are several methodological pitfalls of supermatrix methods that must be considered during the analysis [404]. Additionally, they exhibit practical problems with respect to scalability: Building a multiple alignment and inferring the phylogenetic tree by means of maximum likelihood is resource-intensive, both with respect to human and computational resources.

After the same set of genes is sequenced from all organisms of interest, they are curated so that only those genes that exhibit favorable qualities are taken into account. Genes with an evolutionary history that deviates strongly from the expected species history are labeled as outlier genes [405]; it has been demonstrated that including even a small number of outlier genes in the analysis may significantly alter resulting species trees [406]. The outlier detection is often performed by human experts who screen all gene trees and remove any problematic sequences from subsequent analyses, which constitutes a time intensive step. Furthermore, the time required for the maximum likelihood calculation of a tree involving a large set of genes and species can take several days to weeks of processing time and requires a large amount of memory [236, 407]. Another approach to infer species phylogenies is by means of so-called *supertree* methods [408], such as ASTRAL [409]. Here, the basic idea is to construct a set of phylogenetic trees that correspond to the evolution of the sequenced genes, respectively. Subsequently, all gene trees are merged into a single species tree that represents the overall species phylogeny. In doing so, a fundamental concern is that gene trees are *conflicting*, which means that the species tree cannot adhere to the topology of each gene tree, but instead must resolve the discordance among the gene trees. Different methods are available for the merging of the gene trees; in case of ASTRAL, the merging is performed according to *quartet trees*. A quartet tree is an unrooted phylogenetic tree with exactly four leaf nodes and two inner nodes. A phylogenetic tree with m leaves *induces* $\binom{n}{4}$ distinct quartet trees, in each of which all nodes are removed except for four distinct leaf nodes and two internal nodes such that their relationship in the original tree is retained. ASTRAL finds the species tree that agrees with the largest number of quartet trees induced by the gene trees. Finding this optimal tree is NP-hard, and thus ASTRAL offers heuristics to find non-optimal trees in an acceptable time.

No matter whether supermatrix or supertree methods are used for phylogenetic reconstruction, a newly sequenced species cannot be inserted into an existing tree using the traditional methods for phylogeny reconstruction. Instead, the whole tree reconstruction process must be started from scratch, which poses a substantial bottleneck for the integration of new sequencing data into ongoing projects based on existing phylogenetic trees. Structured placement of new species into existing trees has been attempted only rarely; one example is the custom workflow MGPlacer, which was used to infer the relationship of historical tuberculosis strains from metagenomic data [410]. Thus, it is desirable to streamline the process of inserting new species into existing trees to keep them up-to-date. So far, only two publicly available methods have attempted to systematically insert new species into existing species trees while considering gene discordance. The first method is INSTRAL [411], an extension of the ASTRAL program. In addition to an existing species tree $\mathcal{T}^S_{\text{ref}}$, a query sequence S_q, and corresponding data of all gene sequences of the reference and query sequences, INSTRAL also requires all gene trees as input. INSTRAL then places each gene of S_q on its respective gene tree using any algorithm for phylogenetic placement. Subsequently, as in ASTRAL, it determines the location in $\mathcal{T}^S_{\text{ref}}$ that coincides with most of the quartet trees induced by the updated gene trees that include S_q. Although this procedure accounts for discordance in the gene trees and is statistically consistent, it also requires the storage, handling, and update of all gene trees, and thus, 'complicated analyses pipelines' [412]. INSTRAL cannot only add single species, but also provides two options to add multiple species to an existing tree: Multiple species are added either independently or consecutively, thus resolving their mutual relations. Another recent approach is DEPP [412], which accounts for discordance among gene trees by employing a supervised machine learning approach: DEPP trains a convolutional neural network that embeds the input species in a high-dimensional Euclidean space such that the distances between their sequences is represented. New species can be added to the embedded space even if only a single gene sequence is available. Subsequently, distance-based placement programs may infer the position of the query species in the overall species tree.

An updated version of DEPP uses a hyperbolic space instead of a Euclidean space for the embedding and produces more accurate species trees compared to the original version [413]. The limitations and challenges of the calculation and update of large-scale phylogenetic trees, as well as recent technological advances, have been discussed in detail [414].

5.1.1 Methods

We implemented and evaluated a simple process for the augmentation of existing phylogenetic trees with additional species by means of phylogenetic placement. We assume that there is an *initial* tree to which additional species are added and that the tree was constructed on the basis of multilocus genomic data; we also assume that the gene sequences of all involved species are available. Sequence data for the to-be-placed species are also present as multilocus genomic data. Our proposed method adds these species consecutively to the existing tree, thus augmenting it permanently. For a given query species S_q, our procedure first determines a single placement location for each gene of S_q, and subsequently merges all gene placements into a single species location. This procedure is considerably faster than existing procedures for de-novo tree reconstruction, because phylogenetic placement with App-SpaM is efficiently possible for a large number of genes and species. Furthermore, the augmentation pipeline is straightforward to use, requires little knowledge from the end user, and does not involve additional preprocessing of the sequence data.

More specifically, we receive a set of m species S_i, $i \in \{1, \ldots, m\}$ together with their available sequence data D_i, $i \in \{1, \ldots, m\}$. We assume that the sequencing data for each species comprise a set of o homologous genes. However, the approach can be easily generalized to any reference or query species that comprises only a subset of the o genes. Additionally, a phylogenetic tree T^0 is given whose leaves are labeled from 1 to m, corresponding to the m species. Then, for a new query species S_q and its sequence data D_q, our proposed method inserts S_q into a position of T^0 that is deemed appropriate and outputs the tree T^1 that comprises $m + 1$ species. Subsequently, the procedure is repeated for additional query species to iteratively update T^1, resulting in continuously augmented trees $T^2, T^3, T^4, \ldots$, and so on. For a set $S_Q = \{S_{q_i}, i \in \{1, \ldots, k\}\}$ of k new species, we term the tree that results after inserting all species of S_Q one after the other as T^k.

The location of each single query species S_q is determined with respect to the location of all gene placements of its gene sequences in D_q. The insertion of S_q into T is a three-step process: First, a phylogenetic placement software $\mathcal{P}$ places each gene from S_q into T. Second, the information from all gene placements is merged to infer a single placement position within T. Third, the new species is inserted at the specified position, new branch lengths are determined, and the updated tree with $m + 1$ species is returned. The accuracy of gene and species placements depends on the properties of the organisms under consideration; this includes, in particular, the degree of relatedness between the species in S_Q and the species present in the tree, the size and density of the taxon sampling in T^0, and the evolutionary congruence between single gene trees. For example, if many HGTs occurred between the input species, the positions of gene placements of a single species may be spread across different branches of T. The reason is that the placement position of each horizontally transferred gene is inferred from the species tree topology, which is not congruent with the topology of the according gene trees. In addition to HGTs, other biological mechanisms may cause similar effects that skew the estimation of the species placement position in the second step. For now, we ignore this issue and assume that the resulting placements of HGT-genes simply differ compared to congruent gene placements; we discuss further solutions to this problem in Sec. 5.2. To properly assess the quality of a tree T^k after inserting all species from S_Q, we employ the use of artificially generated data sets. This allows us to control all data set properties individually and to evaluate their influence on the augmented tree. Furthermore, similar to the evaluation of phylogenetic placement, we employ a pruning-based evaluation procedure in order to know the true underlying tree. We create the set S_Q by pruning species from an existing ground-truth tree $\mathcal{T}^*$; then, the pruned species are placed back onto the pruned tree T^0, resulting in T^k. Appropriate metrics are then used to evaluate the accuracy of T^k with respect to $\mathcal{T}^*$ as described below.

We simulated several artificial data sets with varying parameters using the Artificial Life Framework (ALF) [415]. ALF simulates the evolution of a *base genome* consisting of m species along a phylogenetic tree $\mathcal{T}^*$, which is either provided as input to ALF, or generated at random. We use the latter option and let ALF generate a random backbone tree in all experiments. Starting with the base genome at the root of $\mathcal{T}^*$, ALF simulates its evolution along the branches of the tree, following the probabilistic rules of a specified evolutionary model. This process results in one genome for each leaf in $\mathcal{T}^*$. The main simulation parameters of interest are the number of species, the number of genes per species, and the overall mutation rate that determines the degree of relatedness between the genomes. Except for the mutation rate, we do not change any parameters for local-scale evolutionary processes for the simulation of the DNA sequences: By default, ALF uses the TN93 model to simulate substitutions [48], the Zipf distribution for indels which adequately represent insertions and deletions [416, 417], and the Gamma distribution to simulate variation among sites. Furthermore, ALF can also simulate large-scale evolutionary events such as genome rearrangements, gene gain and loss, gene duplication, HGT events, and more. Although we disabled these options by default, we also repeated each simulation with the additional simulation of HGT events. Our Base data set comprises 100 species with 50 genes each, for an average mutation rate of 0.05 substitutions per sequence position. Additional data sets vary with respect to the mutation rate, the number of genes, and the number of organisms; an overview is given in Tab. 5.1.

The pruning-based evaluation procedure takes the true tree $\mathcal{T}^*$, comprising m species and their sequence data, as input. Then, k sequences are selected at random and pruned from $\mathcal{T}^*$, forming the reference tree T^0. After placing all k species, the resulting tree T^k is compared to the original tree $\mathcal{T}^*$ to judge the quality of the augmentation process. There are several methods that compare the similarity between phylogenetic trees, the most ubiquitous being the Robinson-Foulds (RF) metric [418]. For a given tree T, each branch b of T induces a split S_b, that is a distinct bipartition of the leaf nodes into two sets A and B. In a *trivial* split, one of the two sets A or B comprises exactly one leaf. All trivial splits are contained in each tree that is defined on the same set of leaf nodes. Let $S_T = \{S_b, \, b \text{ is branch in } T\}$ be the set of all splits in T. The RF distance d_{RF} between two trees T_1 and T_2 over the same set of leaf nodes is defined as the number of splits that are unique to either T_1 or T_2:

$$d_{\text{RF}}(T_1, T_2) = S_{T_1} \setminus S_{T_2} + S_{T_2} \setminus S_{T_1} \,. \tag{5.1}$$

The RF distance depends on the overall size of the input trees, and larger trees yield higher RF distances on average. *Normalizing* the RF metric, for example, by the maximum possible number of non-trivial splits, overcomes this issue. Although the RF metric is regularly used to quantify the similarity between phylogenetic trees [11, 409, 419], it has been criticized that its values lack a clear meaning and that it exhibits undesired biases such as a fast saturation rate [420]. Even if both input trees vary only with respect to a single leaf node, the RF distance may reach its maximum possible value. Removing the single differing leaf from both trees results in $d_{\text{RF}} = 0$. This problem arises since the RF distance only considers identical splits; the similarity between two near-identical splits is not taken into account by Eq. 5.1. To mitigate these issues, *generalized* RF distances follow an adapted approach: For two splits S_1 and S_2, they define a measure $\psi(S_1, S_2)$ that quantifies the similarity between the splits. For two input trees, they then find a matching

$$M = \{(S_{b_1}, S_{b_2}),\ b_1 \text{ is branch in } T_1,\ b_2 \text{ is branch in } T_2\} \tag{5.2}$$

that assigns each split of one tree to exactly one split of the other tree and vice versa. The overall score $\Psi(M)$ of a matching M is defined as the sum of all matched split similarities:

$$\Psi(M) = \sum_{m \in M} \psi(m)\,. \tag{5.3}$$

The matching with the lowest score is considered to be the overall generalized RF distance d_{gRF} between T_1 and T_2:

$$d_{\mathrm{gRF}}(T_1, T_2) = \min_{M} \Psi(M)\,. \tag{5.4}$$

There are several generalized RF metrics that differ from each other with respect to the definition of $\psi(S_1, S_2)$ [421–423]. Here, we use the clustering information distance (CID) [420], a generalized RF metric which uses concepts from the field of Information Theory to score split similarities between trees: For a split S_1 that implies the two leaf sets A_1 and B_1, let $P(A_1)$ be the probability that a randomly selected leaf belongs to set A_1, thus $P(A_1) = |A_1|/m$, and accordingly $P(B_1) = |B_1|/m$. Likewise, the split S_2 is a bipartition of all m leaves into two sets A_2 and B_2. The entropy of a split S_1 is given as

$$E(S_1) = -P(A_1) \cdot \log P(A_1) - P(B_1) \cdot \log P(B_1)\,. \tag{5.5}$$

Furthermore, let $P(A_1, A_2) = |A_1 \cap A_2|/m$ be the probability that a randomly selected leaf is in both leaf sets A_1 and A_2. Then, the *mutual clustering information* $I_{\mathrm{MCI}}(S_1, S_2)$ between two splits is defined as

$$I_{\mathrm{MCI}}(S_1, S_2) = \sum_{\substack{L_1 \in \{A_1, B_1\} \\ L_2 \in \{A_2, B_2\}}} P(L_1, L_2) \cdot \log\left(\frac{P(L_1, L_2)}{P(L_1)P(L_2)}\right) \tag{5.6}$$

To convert $I_{\mathrm{MCI}}(S_1, S_2)$ into a distance measure $\psi_{\mathrm{MCI}}(S_1, S_2)$ it is subtracted from the total entropy of all splits in both trees divided by two [420].

We infer a placement position for S_q via one of three different approaches. The first approach simply calculates a singular placement branch based on the complete data in D_q using any program for phylogenetic placement, instead of placing each gene of D_q individually. We term this approach the *collective* species placement. The other two approaches first infer placements for each individual gene in D_q, and then find a placement for the species S_q based on the distribution of the gene placements across the branches of the tree. More specifically, the second approach is a top-down tree traversal similar to the placement heuristics proposed in Sec. 4.2. We assign a weight $w(b)$ to each branch b of the reference tree that is equivalent to the total sum of gene placements occurring at b or in the subtree below b. We then follow the highest weighted path from the root of the tree towards the leaves. Let u and v be the two branches below b; the traversal is either stopped when reaching a branch above a leaf node, or when

$$\otimes_1 = |w(u) - w(v)| > \frac{w(b)}{2} \tag{5.7}$$

Table 5.1 – Simulated data sets for the evaluation of continuous tree augmentation.

abbreviation	# organisms	# genes	mutation rate	HGT rate
BASE	100	50	0.05	0 / 0.5 /
HIGH-MR	100	50	0.01	0 / 0.5
MANY-GENES	100	100	0.05	0 / 0.5
MANY-SPECIES	200	50	0.05	0 / 0.5

We performed experiments on four data sets that differ with respect to the number of organisms, number of genes, and applied mutation rate. Each data set was simulated once without HGT events and once with HGT events of the specified rate. The first column indicates the name for each data set.

holds; we term this method the *tree traversal* species placement. The third approach is called the *clustering* method for species placement: Here, we segment the branches of the tree into clusters according to their corresponding gene placements and use the *centroid* branch of the largest cluster as the species placement location. Again, we define the weight $w(b)$ of a branch b to be the number of gene placements that occur on b (this differs from to the tree traversal method where the weight *sums up* all placements *below* branch b). We define two branches b_1 and b_2 to be adjacent when the levels of their attached parental nodes differ by exactly 1. A group of l branches $C = \{b_i,\, i \in [l]\}$ forms a single cluster if $w(b_i) > 0,\, i \in [l]$ applies and if for any two branches $b_{i_1}, b_{i_2} \in C$ a chain of adjacent edges $b_{i_1}, \ldots, b_j, \ldots, b_{i_2}$, $b_j \in C$ exists. According to these rules, we group the branches of the tree into unique clusters based on their weights. Given a cluster C, the centrality $c(b_i)$ of a branch $b_i \in C$ is defined as the total amount of force required to transfer the weights from all other branches b_j, $j \neq i, j \in [l]$ within C to b_i. Here, the force is defined as the weight of b_j times the number of nodes between b_i and b_j. The centroid b_C of a placement cluster C is defined as

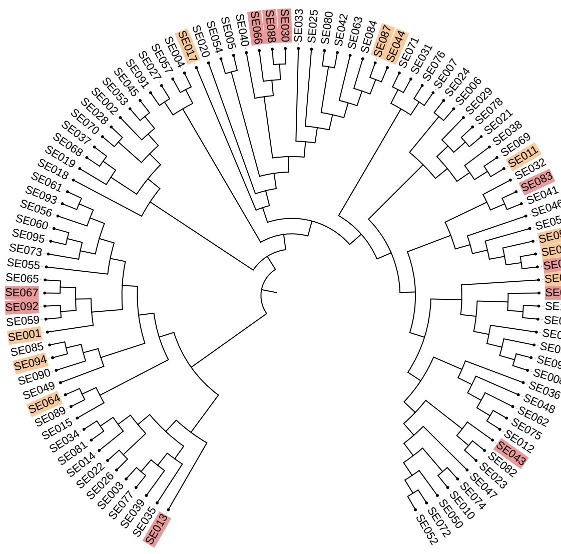
$$b_C = \underset{b \in C}{\arg\min}\, c(b)\,. \tag{5.8}$$

The clustering method chooses the centroid of the largest cluster as the overall species placement location for the query species S_q.

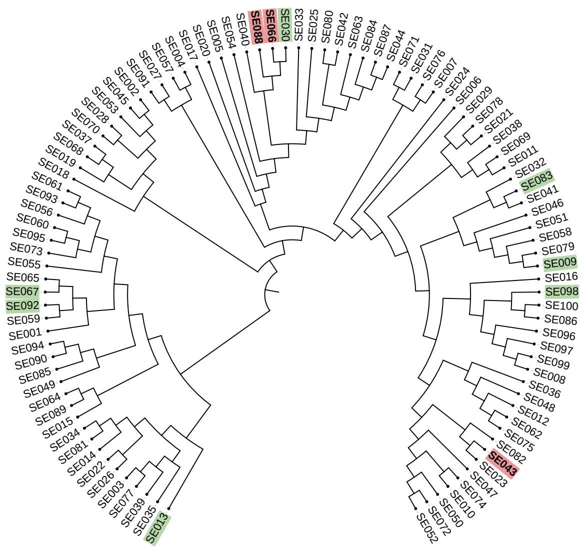
Although this process may be applied for any phylogenetic placement program $\mathcal{P}$, APP-SPAM is particularly suited due to its fast placement speed and applicability in the absence of gene alignments. In addition to APP-SPAM, we also used EPA-NG in order to compare the achieved results. When using EPA-NG, we calculate an alignment for each gene of S_q with the software MAFFT, concatenate all gene alignments, and infer a single placement position for the aligned sequence. This leads to better placement results than merging individual gene placements with one of the heuristics described above. For each of the data sets given in Tab. 5.1, we performed pruning experiments with ten pruned sequences. For each experiment, we measured the accuracy of all proposed methods as the CID distance from the original tree over five repetitions. Additionally, we conducted experiments with 20 and 50 pruned reference sequences for APP-SPAM, respectively. We also tested the influence of the parameter of APP-SPAM's default SPAM-X heuristic and ran additional experiments with a larger number of HGT events.

5.1.2 Results

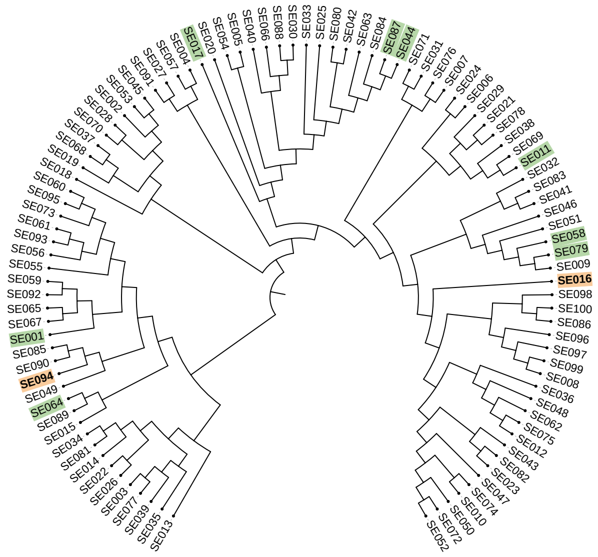
Due to the stochastic nature of the data set simulation in our analysis pipeline, different sequences were pruned for the evaluation of App-SpaM and EPA-ng. We expect that any differences that might improve or worsen the augmentation process balance out over the five



(a) Simulated ground truth tree.



(b) Augmented tree with App-SpaM.



(c) Augmented tree with EPA-ng.

Figure 5.1 – Exemplary phylogenetic trees augmented with new species by App-SpaM or EPA-ng. Part (a) shows the true underlying tree as simulated with ALF. The sequences to be pruned are highlighted in color for App-SpaM (red) and EPA-ng (orange). Part (b) shows the augmented tree after placing the pruned sequences with App-SpaM and the clustering method. The augmented sequences are highlighted in green, unless they were incorrectly placed (red, bold font). Part (c) shows the augmented tree after placing the ten pruned sequences with EPA-ng. The augmented sequences are highlighted in green, unless they were incorrectly placed (orange, bold font).

repetitions of each experimental setup. Figure 5.1 shows the simulated tree $\mathcal{T}^*$ of the first repetition for the Base data set as an example. In addition, two trees are presented that were augmented with App-SpaM and the clustering method, and EPA-ng, respectively. In this specific example, App-SpaM places 7 of the 10 pruned species at their correct location, while EPA-ng places 8 of the 10 species correctly. All incorrectly placed species deviate from their original position by exactly one node for both App-SpaM and EPA-ng. On visual inspection, most other test cases exhibit similar properties with respect to the ratio of correctly placed sequences as well as their degree of accuracy. For App-SpaM, two of the wrongly placed sequences belong to the same clade of three pruned sequences; thus, the sequences were placed one after another, utilizing branches that were added in previous iterations.

Figure 5.2 summarizes the CIDs between the reconstructed trees and the respective original trees on all data sets shown in Tab. 5.1, either with or without simulated HGT events. Again, due to the nature of the stochastic simulation, the number of HGT events varies between the experimental setups: For the data set Base, 52 genes out of 5 000 originate from an HGT rather than from inheritance. Accordingly, High-Mut has 76, Many-Genes has 101, and Many-Species has 55 HGTs. EPA-ng was executed with default parameters on the concatenated query genes, resulting in a single *collective* placement position.

In general, all trees augmented with EPA-ng display a high similarity to the original trees, regardless of the presence of HGTs. In 30% of cases, the augmented tree topology is identical to the true topology. The variation of the CID across the five repetitions remains below 0.02 for all data sets processed by EPA-ng. In contrast, trees augmented with App-SpaM have a CID that is twice to three times as high. We proposed three methods to merge individual gene placements to a single species placement position, namely, the collective method, the tree traversal approach, and the clustering procedure. All augmented trees illustrate similar deviations from true trees, no matter which of the three methods was employed. Similarly to EPA-ng, the presence of HGT events does not have a significant influence on the results of App-SpaM. For both programs, we observe a drop in the CID for the Many-Species data set; however, the decline of the CID is caused by the increase in tree size rather than improvements in the quality of augmented trees. App-SpaM was executed with default parameters and used the SpaM-X heuristic with $X = 4$.

Using different values for X drastically changes the accuracy of the augmented trees, as seen in Fig. 5.3. Using $X = 500$, thus shifting gene placements toward more distal leaves, substantially improves the quality of augmented trees: The CID drops by at least 50% for all four data sets. When more than 10% of species are pruned from the reference trees, the CID increases roughly linear with the number of species for App-SpaM, see Suppl. Fig. C.2. Increasing the number of HGT events by a factor of four has a significant influence onto the placement accuracy as illustrated in Suppl. Fig. C.1. One possible solution to this problem is presented in the next section. App-SpaM performed the consecutive augmentation of 10 species on the Base data set—comprising 90 species with 50 genes each—on an Intel(R) Core(TM) i5-2500K CPU with 3.30 GHz and 8 GB RAM in 149 seconds (average over five repeats). On the same system, EPA-ng did not finish due to the limited RAM; instead, we ran EPA-ng on our local computing cluster comprising 30 Intel(R) Xeon(R) E7-4850 CPUs with 2 GHz and 1 TB RAM where it required 4 699 seconds (average over five repeats).

5.1.3 Discussion

We presented a straightforward procedure to iteratively update phylogenetic species trees with additional organisms, a process that we also refer to as *tree augmentation.* We used three

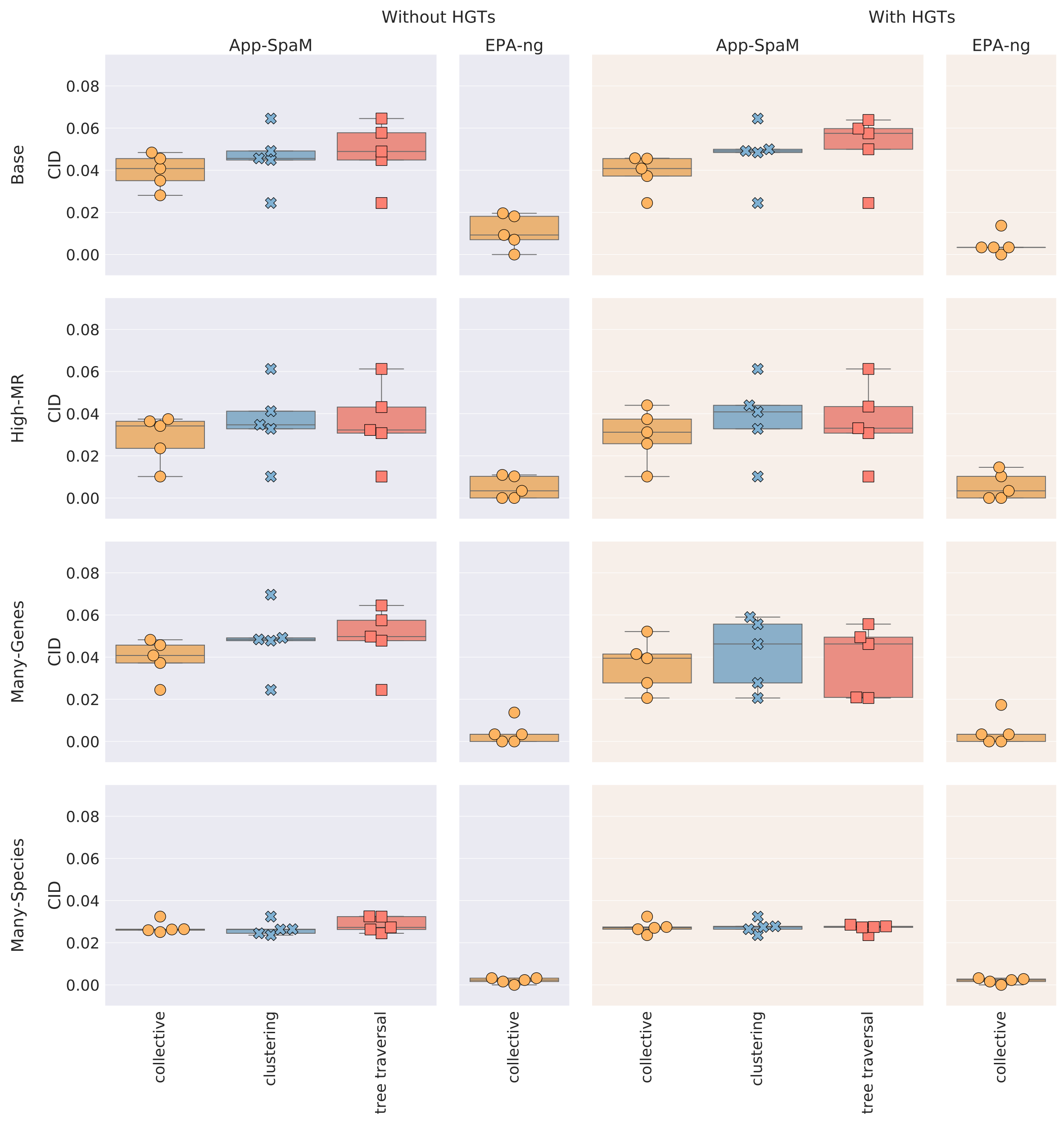


Figure 5.2 – Accuracy of tree augmentation using App-SpaM and EPA-ng on four artificial data sets without (left side, blue background) and with (right side, orange background) simulated HGT events. Roughly 1% to 2% of genes originate from an HGT when the HGT simulation is turned on. All three methods to infer species placement positions are presented for App-SpaM. From left to right, these methods are the collective placement (orange), the clustering approach (blue), and the tree traversal method (red). For EPA-ng only the collective placement (orange) was performed. Additional results for App-SpaM are given in Suppl. Fig. C.3 and Suppl. Fig. C.4 which show that App-SpaM's accuracy improves when using $X = 500$ instead of its default version.

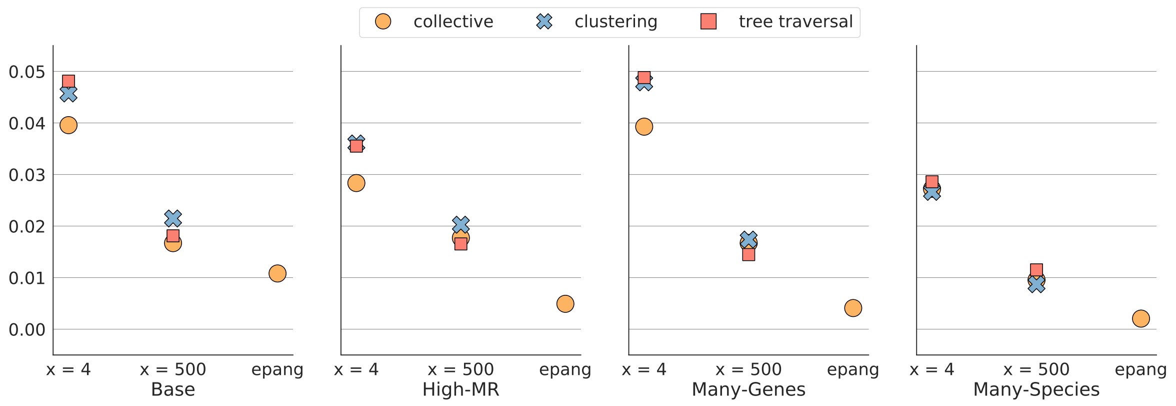


Figure 5.3 – Mean accuracy of tree augmentation using App-SpaM and EPA-ng on four artificial data sets without simulated HGT events. For App-SpaM, two different values ($x = 4$ and $x = 500$) are shown for the parameter x of the SpaM-X heuristic for each of the three consensus methods. For EPA-ng, we only used the default parameters with the collective species placement method.

techniques to find appropriate species placements: The first approach simply uses existing placement software on the concatenated sequence data to directly determine a single species placement. For the latter two approaches, first, each gene of a query S_q is placed onto the existing tree with common placement programs and, second, all gene placements are merged to a single species placement location using a tree traversal or clustering approach. Neither of these approaches explicitly accounts for the potential discordance between genes of S_q; instead we assume that the consensus of all genes in a query resolves any potential conflicts and approximates the correct species position sufficiently accurate. When genes are placed with EPA-ng, the augmented trees are of high quality and near identical to correct trees. However, trees augmented by App-SpaM with default parameters are considerably less accurate than those augmented by EPA-ng with respect to the CID. Using non-default parameters for App-SpaM improves the quality of the augmented trees considerably. Incorrectly placed species are always placed close to their correct position for both placement programs. The moderate occurrence of HGT events does not influence the accuracy of augmented trees, but the quality of augmented trees deteriorates when an increased number of HGTs is simulated. Although the augmentation accuracy with App-SpaM requires further improvement, it only requires a fraction of the time compared to using EPA-ng. Even for small data sets, it was not possible to execute EPA-ng on a current standard personal computer.

The results presented here are not in direct agreement with the results presented in Sec. 3.1.2. There, we demonstrated that the accuracy of App-SpaM is comparable to alignment-based placement programs when placing short read sequences from metagenomic experiments. Multiple reasons are conceivable for why EPA-ng outperforms App-SpaM in the task of tree augmentation. One reason could be the length difference of the query sequences; when read lengths are increased from 150 bp to 300 bp, we already observed that alignment-based programs have an enhanced accuracy compared to App-SpaM, see Fig. 3.10. Using even longer sequences, such as whole genes, could further amplify this effect and may result in the difference in precision that we observed in Fig. 5.2. Another potential reason may be the setup of our evaluation procedure: Including a larger variety of data sets with respect to their size, sequence similarity, and tree structure might yield an explanation for our observations. Especially the strong improvement of the augmented trees when increasing the SpaM-X parameter to $X = 500$ suggests that the relative positions of pruned sequences impact the quality of resulting trees: With higher values of X, gene placements are shifted toward the leaves of the reference tree. Consequently, if species are only pruned from distal branches of the reference tree, the placement accuracy of genes might improve. Note, however, that we did investigate this in Sec. 3.1.3 and that we did not observe such an effect for short read sequences. The true underlying reason for the observed results remains unclear for now; in the future, a structured analysis should be performed that focuses on the number and relative location of the pruned sequences to assess the influence of the proposed effects, both for App-SpaM and for other placement tools.

Two other software packages—INSTRAL and DEPP—have recently been introduced for the update of large phylogenetic trees. Neither of these tools were publicly available when we conducted our analyses. INSTRAL places the genes of S_q onto their respective gene trees and uses the consensus among derived quartet trees to find a species placement position. DEPP utilizes deep learning to find species placements even when only a few genes or a single gene of S_q are available. Similarly to our evaluation procedure, INSTRAL has been compared with EPA-ng for the placement of single species, or the consecutive placement of multiple species; it has been reported that INSTRAL outperforms EPA-ng, especially when the discordance among gene trees is high [411]. DEPP has been compared to INSTRAL, EPA-ng, and APPLES with the contradictory result that DEPP and EPA-ng are on par and generally perform better than INSTRAL and APPLES, especially if new species are placed on the basis of only a few genes [412]. These divergent results are probably explained by the differences in the evaluation pipelines. Only when species positions are inferred from many genes, the precision of INSTRAL is comparable with other programs. Because our results show that App-SpaM does not reach the accuracy of EPA-ng, it is evident that our current procedure for the iterative augmentation does not produce phylogenetic trees as accurate as those from INSTRAL or DEPP. However, our pipeline has the advantage of being easy to use and fast to execute. Furthermore, it can be based on any phylogenetic placement program other than App-SpaM and EPA-ng. In general, the proposed method should be considered as a general boosting technique that broadens the applicability of existing placement programs rather than an individual software program.

Regardless of the reported results, discrepancies in evaluation procedures among different approaches remain an issue and prohibit meaningful comparison. While we used ALF to simulate artificial data sets, DEPP and INSTRAL employed the SimPhy software [424] and simulated incomplete lineage sorting instead of horizontal gene transfer to model gene tree discordance. Furthermore, the evaluation of DEPP and INSTRAL utilizes the original RF distance to calculate tree dissimilarities instead of generalized RF distance measures. Additionally, all evaluation pipelines only consider the topology of augmented trees and not the branch lengths of newly added species. These are just a few of the many differences between existing evaluation procedures; it would be highly beneficial for future research efforts to develop a common framework for the evaluation of tree augmentation programs, similar to the PEWO framework for phylogenetic placement. Such a framework should include multiple workflows that assess different aspects of the augmentation task, such as independent and consecutive augmentation, or metrics that include branch lengths versus metrics that only consider tree topologies. Furthermore, no systematic studies exist that compare how accurate the update of phylogenetic trees is compared to de-novo tree reconstruction. INSTRAL reports that there is decent loss in accuracy when performing an iterative update of existing trees compared to applying de-novo reconstruction with ASTRAL, but it is not clear under which conditions and to what extent these discrepancies occur.

Another factor in this regard may be the order in which species are added to the tree. So far, all programs performed the iterative augmentation in a random order. Although there ought to be no dependency between the augmentation order and the quality of resulting trees in the optimal case, it is conceivable that the order influences the results after all: Inserting 'difficult' species with a high placement uncertainty as late as possible ensures that additional information from previously placed species is available on the basis of which the placement may be improved. We performed a brief analysis of this idea—as explained in detail in Suppl. Chpt. C—but discovered no such effect for App-SpaM, see Suppl. Fig. C.5.

5.2 Gene and Species Outlier Detection

Gene discordance is the disparity in evolutionary history among genes that originate from the same organism. Discordance between genes originates from a variety of different mechanisms and has a major impact on phylogeny reconstruction [425–427]. Correspondingly, it also influences tree augmentation on the basis of multi-locus data as performed in the preceding section. While our results indicate that the simulation of a few HGTs does not affect the quality of updated trees, the presence of many HGTs does worsen the results substantially, see Suppl. Fig. C.1. Similarly, incomplete lineage sorting affects the update of phylogenetic trees [411]. During tree augmentation, INSTRAL accounts for gene discordance by placing each gene on its individual gene tree and merging the placements via the combined information of quartet trees. However, this procedure requires the continuous storage and maintenance of all gene trees, which can pose a substantial practical hurdle, especially if many genes are available. In contrast, the program DEPP proposed a deep learning framework to update species trees even on the basis of single genes. Similarly, it is common practice to remove genes or species with challenging evolutionary histories prior to de-novo tree reconstruction, and case studies have demonstrated that the removal substantially improves resulting tree topologies [428]. According genes or species are also called *outlier genes* or *outlier species*. However, the identification of these genes often involves a time-consuming manual inspection of all gene trees. Only a few methods exist that detect and remove outliers automatically. One such method is Phylo-MCOA, which identifies both gene and species outliers by a simultaneous inspection of all gene tree topologies using multiple co-inertia analysis [405]. Likewise, we evaluated whether alignment-free programs are suitable to detect gene or species outliers prior to de-novo phylogeny reconstruction in earlier work [429]. Given a dataset of m sequences with l genes each, we constructed a $m \times m/2$ dimensional vector of pairwise distances for each of the l loci using Read-SpaM or FSWM. Then, we reconstructed all l gene trees with NJ based on the distance estimates and calculated pairwise CIDs between all gene trees. To flag outlier genes, we either used statistical measures based on the pairwise gene tree distances, or the clustering algorithm DBSCAN [430] on the vectorized between-loci distances. The former approach is inspired by the analysis of tree spaces with multidimensional scaling and the effect of different tree-to-tree distances on tree spaces [431, 432]. The advantage of using alignment-free methods is their fast computation speed, which allows the analysis of large amounts of genes in little time. On the downside, the topologies of the resulting gene trees are generally less reliable than *ML*-based trees which renders the detection of spurious gene trees more difficult.

Here, we transfer our approach to gene or species outlier detection from de-novo reconstruction to the augmentation of existing trees. In contrast to INSTRAL, our approach does not need to store and update individual gene trees. The tree augmentation procedure proposed in Sec. 5.1 involved an *implicit* integration of outlier genes by using consensus criteria among all gene placements. While this worked for a low number of HGT events, the accuracy decreased sharply when more HGTs were present, see Suppl. Fig. C.1. Therefore, we extend our approach from Sec. 5.1 by *explicitly* identifying and removing outlier genes or species and evaluate whether the tailored approach produces improved results. We treat outlier detection as a binary classification process where each gene or species either qualifies or does not qualify as an outlier. A classification method for outliers assigns a label to each gene that indicates its membership to one of the two classes. To identify and remove spurious genes or species, we performed a pre-placement step similar to the ordered placement that we analyzed in Suppl. Sec. C.1. During pre-placement, each gene of each query species S_q is independently placed onto T^0 and information about individual gene placements is gathered for each species. More specifically, we calculate a set of features for each gene of each species, as well as an independent set of features for each species. The outlier detection is performed based on these feature matrices. The set of gene features comprises its placement level in T^0, its average distance to all other gene placements, information about which reference sequences are contained in the subtree below its placement position, the number of SpaMs to each reference sequence, the estimated distances to each reference sequence, and information about the relative deviation of its placement compared to genes of the same loci in other organisms. The set of species features comprises information on the distribution of its gene placements among T^0, the variation of the number of SpaMs to each reference sequence among its genes, and the variation of the distance estimates to each reference sequence among its genes. When T^0 comprises m species, the total number of features for each gene and species is $3 \cdot m + 4$ and $8 \cdot m + 5$, respectively. A detailed overview of all features and their derivation is given in Suppl. Sec. C.2. After pre-placement and feature creation, one feature matrix contains all to-be-placed species, and additionally single feature matrices contain all genes for each to-be-placed species. We applied Isolation Forests to each feature matrix individually to label genes or species as outliers. All identified gene or species outliers are removed from the data and placement is performed with the remaining ones.

To evaluate our method, we repeated our experimental setup from Subsec. 5.1.1: We simulated five artificial data sets using ALF with 100 species with 50 genes each with an average mutation rate of 0.05 substitutions per sequence position. For another five artificial data sets, we additionally turned on the simulation of HGT events with the high rate that led to deteriorated results in the previous experiments. With the specified rate, we observe that between 4% and 8% of all 5 000 genes in each data set originate from HGTs. For each experiment, we prune ten random organisms and place them back onto the tree with all three proposed augmentation heuristics. The placement is performed once without the removal of outliers, once with the removal of gene outliers, and once with the removal of gene and species outliers. For the outlier detection, we ran an Isolation Forest on each feature matrix using the SCIKIT-LEARN [433] implementation with default parameters. We compare all resulting trees with the known correct trees using the CID metric. To assess the precision of the resulting labels, we assume that each horizontally transferred gene qualifies as an outlier. We do not take into account between which species the gene transfer occurred. We also do not explicitly simulate species outliers; implicitly, those species for which many genes were transferred horizontally are expected to be more difficult to place.

Figure 5.4 shows how the quality of augmented phylogenetic trees depends on the removal of outlier genes or species. As before, a higher rate of HGTs strongly reduces the quality of augmented trees, and the CID rises significantly compared to data sets without simulated HGT events. Removing all outlier genes with our proposed procedure improves the accuracy substantially for all three methods and heuristics, but trees do not attain the same quality

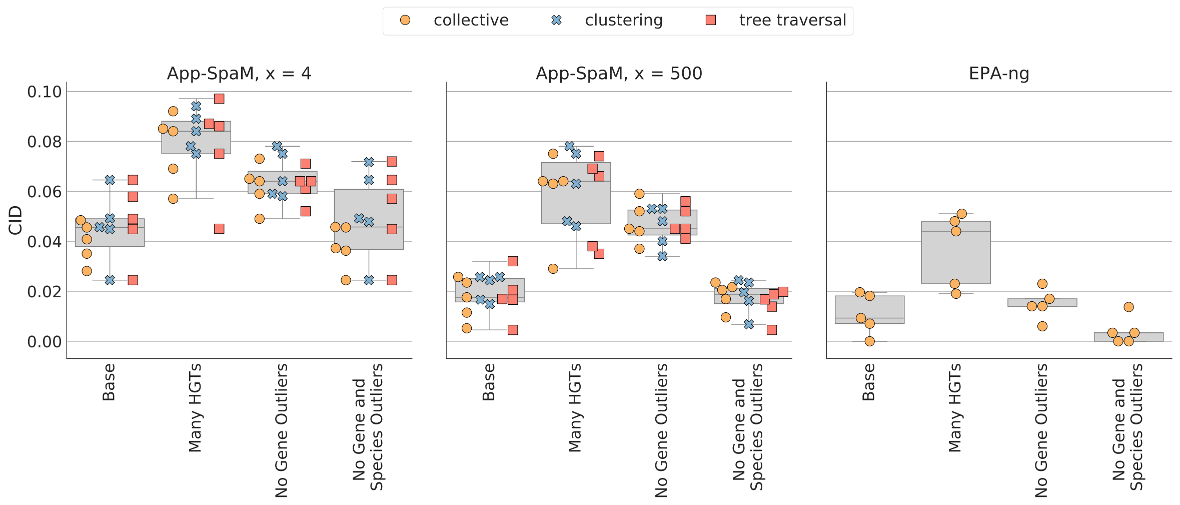


Figure 5.4 – Accuracy of tree augmentation with and without outlier detection. Each plot (left, middle, right) shows the accuracy of augmented trees with a different program (App-SpaM with $X = 4$ on the left, App-SpaM with $X = 500$ in the middle, EPA-ng on the right). The accuracy is measured by the clustering information distances (CID, y-axes). For each program, we compare the results for a data set without simulation of HGTs (Base), and with the simulation of HGTs. In the latter case, we compare augmented trees without outlier detection (Many HGTs), with gene outlier detection (No Gene Outliers), and with gene and species outlier detection (No Gene and Species Outliers). As before, we use all three augmentation heuristics for App-SpaM (differently colored markers).

as without HGTs. If gene and species outliers are removed, resulting trees are consistent with those augmented trees where no HGTs were present. Again, this observation holds for App-SpaM for all three species placement heuristics as well as for EPA-ng. Note, however, that the trees in the last scenario do not include all 100 species; after removing species outliers, an average of 94.4 species remain. The average accuracy of inferred labels is 95.34% (SD 1.75) with a recall of 93.92% (SD 2.34). Thus, most of the true outliers are detected by our approach; see the exemplary confusion matrices in Fig. 5.5. At the expense of the high recall, the average precision reaches only 60.18% (SD 5.25). This means that a large portion of genes flagged as outliers are genes which were not transferred horizontally. Figure 5.6 is a simplified visualization of our gene feature space including 50 genes of a single species in a single experiment. For this particular species, four out of five outlier genes were recognized correctly. Another four genes were incorrectly labeled as outliers although they were not transferred horizontally between organisms.

Our presented study implies that the presence of discordant gene histories impacts the augmentation of existing phylogenetic trees. As a proof of concept, we devised, implemented, and evaluated a process to detect and remove outlier genes or species based on the placement data of a pre-placement step of the to-be-added species. Augmenting trees after removing gene and species outliers results in significantly better quality. In contrast, removing only gene outliers improves the trees, but not to the same extent. As suggested in an earlier analysis, this effect only applies when sufficiently large gene discordance is present; the presence of only a few HGT events does not significantly influence the proposed heuristics for tree augmentation. On average, our approach detects and removes most outlier genes at the cost of also removing some non-outlier genes; however, the false-positive rate stays below 5% on average, meaning

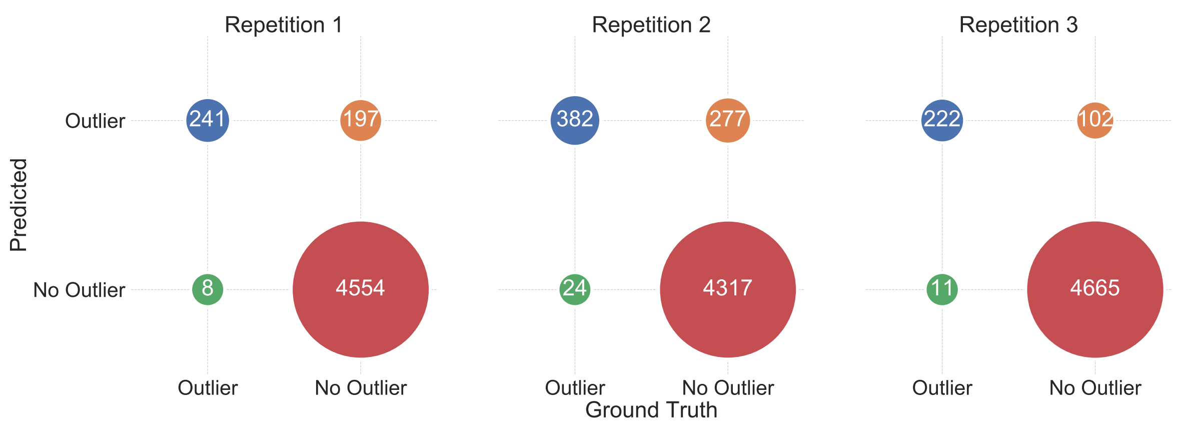


Figure 5.5 – Confusion matrices for detecting gene outliers in three exemplary experiments. Each confusion matrix shows the number of correctly and incorrectly detected outlier genes and non-outlier genes (one circle each, sizes correspond to the displayed counts).

that only a fraction of the overall gene content is discarded for the benefit of removing most outliers. If the proportion of outlier genes is further increased, it might become necessary to find a better balance between recall and precision.

We performed anomaly detection with the implementation of SCIKIT-LEARN's Isolation Forests with default parameters on the basis of a broad selection of features for genes and species, respectively. The features represent how genes were placed in the pre-placement, the distance and spaced-word statistics from spaced-word matches, and varying metrics that describe the consensus among gene placements. The *contamination* parameter of Isolation Forests fundamentally drives the amount of samples that will be labeled as outliers. A higher ratio of samples is labeled as outliers when raising the parameter, while lowering it will reduce the number. This affects the trade-off between precision and recall and should be taken into account accordingly in the future. Tuning the contamination parameter to the data set at hand, potentially with prior knowledge of the expected proportion of outlier genes, could further improve the classification process. Furthermore, a multitude of other advanced anomaly detection algorithms exist and may be used instead of Isolation Forests [434]. The performance of different algorithms also depends on the number and nature of features for both genes and species, as feature selection plays an important role in outlier detection [435]. Our feature space simply represents a first draft and is neither polished nor exhaustive. Discarding futile features or supplementing additional ones that encode contrasting properties could additionally improve the predictive capacity.

Our results presented here are in agreement with other studies that demonstrate the impact of gene outliers on the update of phylogenetic trees [428]. However, sufficiently many HGTs must be present before their negative influence is observable in our experiments. Besides HGTs, the simulation of other biological mechanisms, such as incomplete lineage sorting, may have additional impact on the augmentation of phylogenetic trees. In future work, including and comparing a wider array of mechanisms that result in gene discordance is necessary to assess whether our methods lead to comparable results in more general settings. We also simplified the problem by treating outlier detection as a binary classification problem: We assumed that each horizontally transferred gene is an outlier. However, the donor and target organisms between which the HGT occurred dictate the influence of the gene on the overall tree augmentation or de-novo reconstruction. Assessing the magnitude of the discordance

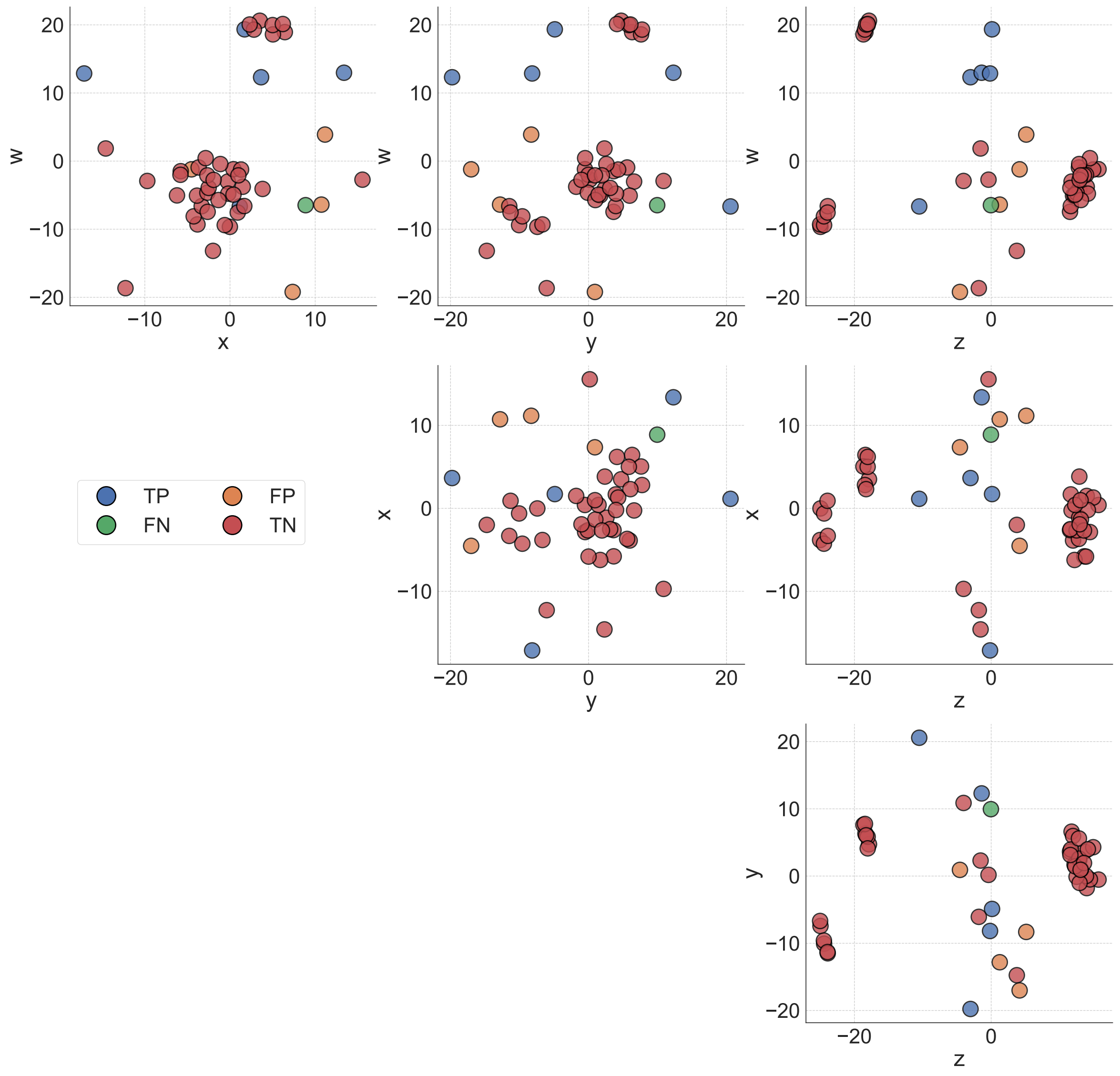


Figure 5.6 – Gene outlier detection for a single species visualized with multidimensional scaling (MDS). We applied MDS to reduce the gene feature matrix of a single organism to a 4-dimensional Cartesian space (dimensions w, x, y, z are annotated at each axes). Each gene of the species is represented by a single point (50 genes in total). Each gene originates from an HGT event (TP and FN) or from inheritance (TN and FP). Genes that originate from an HGT and were correctly labeled as an outlier are referred to as a true positive (TP, blue dots). Genes that originate from an HGT and were not labeled as an outlier are referred to as false negative (FN, green dot). Accordingly, correctly identified regular genes are true negatives (TN, red dots), and regular genes incorrectly labeled as outliers are false positives (FP, orange dots). One such plot may be created for the genes of each organism.

might shed further light on the intricacies of our method: HGTs that result in low discordance are presumably more difficult to classify unambiguously. Furthermore, although we performed species outlier removal, we did not explicitly simulate species outliers in the first place. Future tests for the detection of species outliers could simulate reticulate evolution to derive species for which a non-tree-like history is present. Still, we believe that our analysis showcases that discordance among genes negatively influences the augmentation of existing phylogenetic species trees. This especially holds true for procedures that use a consensus among gene placements such as the one presented in Sec. 5.1, but probably also applies for any other method, similar as in de-novo tree reconstruction. The presented procedure quickly assesses and discards HGT genes, and augmentation results improve considerably for those data sets where HGTs are removed. App-SpaM is also ideally suited in this case, because the necessary pre-placement step, as well as the handling of potentially many long gene sequences, is too computationally expensive for ML-based placement programs.

Chapter 6

Side Projects

> Rebecca Fee Dietlinde Regendantz (Department of Bioinformatics, Georg-August-Universität Göttingen) and Matthias Blanke jointly conceived, planned, and executed all experiments in Sec. 6.1 [436]. Jakob Störiko (Institute of Computer Science, Georg-August-Universität Göttingen), implemented and provided the algorithm utilized in Sec. 6.2. Matthias Blanke conceived, implemented, and executed all experiments in Sec. 6.2.

We outlined common tasks in the fields of phylogenomics and metagenomics in Sec. 2.6 and specifically addressed the problem of phylogenetic placement by presenting a novel algorithmic approach in Chpt. 3 and its potential applications in Chpt. 5. We evaluated our method App-SpaM on simulated and real-world data sets for different applications to assess its accuracy and robustness; although its placement accuracy is of high quality, we observed that distance estimates derived from spaced-word matches sometimes deviate from expected distances, see Sec. 4.3. The magnitude of this deviation seems to be mainly influenced by the underlying sequence data. Here, we further investigate this interrelation for the task of phylogeny reconstruction on an intricate real-world data set using another spaced-word matches approach: In Sec. 6.1, we apply the alignment-free method Read-SpaM to reconstruct a phylogenetic tree of eight Old World monkeys from ancient DNA. In doing so, we specifically examine the accuracy of Read-SpaM's distance estimates and their effect on inferred phylogenetic trees. While it has been demonstrated that spaced words can be used successfully to infer phylogenetic trees [11], our use case presented here is unique with respect to its noisy eukaryotic sequencing data compiled from museum exhibits.

We also restrained our underlying methodological approach to the exclusive use of spaced-word matches so far. In general, most alignment-free methods are based on continuous or non-continuous words of a *fixed* length [137]. However, other properties of molecular sequencing data may also offer information on evolutionary relationships without relying on multiple sequence alignments. For example, it has been recently suggested that insertions and deletions provide information on the similarity of sequences [437]. Finding new sources of information that characterize the degree of relatedness between molecular sequences is indispensable to promote the field of alignment-free methods. New approaches may provide sufficient information to be used alone, or they can be used together with existing approaches to jointly improve distance estimation. We study the use of an alignment-free measure that is inferred from *subsequences* of arbitrary length, the Simon's congruence, in Sec. 6.2.

6.1 A Phylogeny of Old World Monkeys

Spaced-word approaches have been shown to accurately estimate distances from simulated and real-world data [11]. However, the majority of data sets comprised sequences of prokaryotic origin. Prokaryotic DNA is predominantly made up of functional regions and is comparably short with a maximal length of 12 mega base pairs [438]; in contrast, eukaryotic DNA is generally several magnitudes longer and more complex than eukaryotic DNA. The relative amount of intron-exon structures in eukaryotic DNA is rather low, and the total fraction of protein-encoding exons is estimated to be only approximately 1.9% [439]. The majority of eukaryotic DNA consists of *repetitive* regions, such as long terminal repeats, interspaced nuclear elements, transposable elements, or satellite DNA. For example, the proportion of repetitive regions in human DNA is estimated to be between 50% [440] and 66% [441]. Moreover, the degree of conservation varies strongly between different non-coding regions in human DNA [442].

The program Read-SpaM is a reimplementation of FSWM and uses filtered spaced-word matches to estimate phylogenetic distances between *bags of reads* [166]. Here, a 'bag of reads' refers to a loose collection of short sequencing reads that belong to the same organism; subsequently, we simply use the word *sequence* to also refer to a bag of reads. Just like FSWM, Read-SpaM considers each pair of input sequences individually and extracts all spaced words from the two target sequences. Then, it creates spaced-word matches using the same greedy procedure as in FSWM and discards all matches with a score below the predefined threshold. Phylogenetic distances are estimated from the don't care positions of the remaining spaced-word matches by applying the Jukes-Cantor formula. For prokaryotic sequences, the resulting distance estimates are of high quality, even for very low coverage of the input reads. However, the underlying experiments were carried out in a controlled environment, either between a real *E. coli* genome and an artificially modified version of it, or between two different *E. coli* strains. The accuracy of the distance estimates was not evaluated for eukaryotic genomes, although whole genome shotgun sequencing methods commonly produce short reads that originate from any region of the target genome, as discussed in Sec. 2.6. In the context of our observations regarding inaccurate distance estimates of App-SpaM, it remains unclear under which preconditions SpaM approaches produce accurate distance estimates in general; especially in the presence of highly complex eukaryotic data sets from low-coverage whole genome sequencing experiments, distance estimates likely depend on the composition of genetic regions present within the target genomes.

Thus, as an exemplary study, we applied Read-SpaM to a collection of molecular sequences of eight Old World monkeys, taxonomically designated as *Cercopithecidae*. Sequencing data was derived from up to 150 years old museum exhibits. The raw paired-end sequences from whole-genome shotgun sequencing were provided to us as unassembled forward and backward reads. Due to the nature of the samples, the sequencing reads achieve coverages of merely 0.1X to 0.4X of the underlying primate genomes. This made it impossible to create longer contigs or assemblies of the nuclear genomes and no phylogenetic analysis could be conducted using conventional ML or MP methods based on the nuclear genome. Instead, we attempted to infer the phylogenetic relationships of the eight species by processing the provided data set with Read-SpaM and NJ. The two major goals of this study were to assess the accuracy of Read-SpaM's distance estimates when challenged with sequencing data of questionable quality, and to analyze the accuracy of inferred phylogenetic trees.

In the following, we provide a brief summary of our methodological approach; a complete overview is given in previous work [436]. We first applied Trimmomatic [284] to perform basic quality control and trimming of the reads. We did not merge paired-end reads, but instead used the joint collection of forward and backward reads of each organism as input for Read-SpaM. To this end, we extended Read-SpaM to consider only the canonical version of each spaced word, see also Subsec. 3.1.3. Although the reads in this data set have

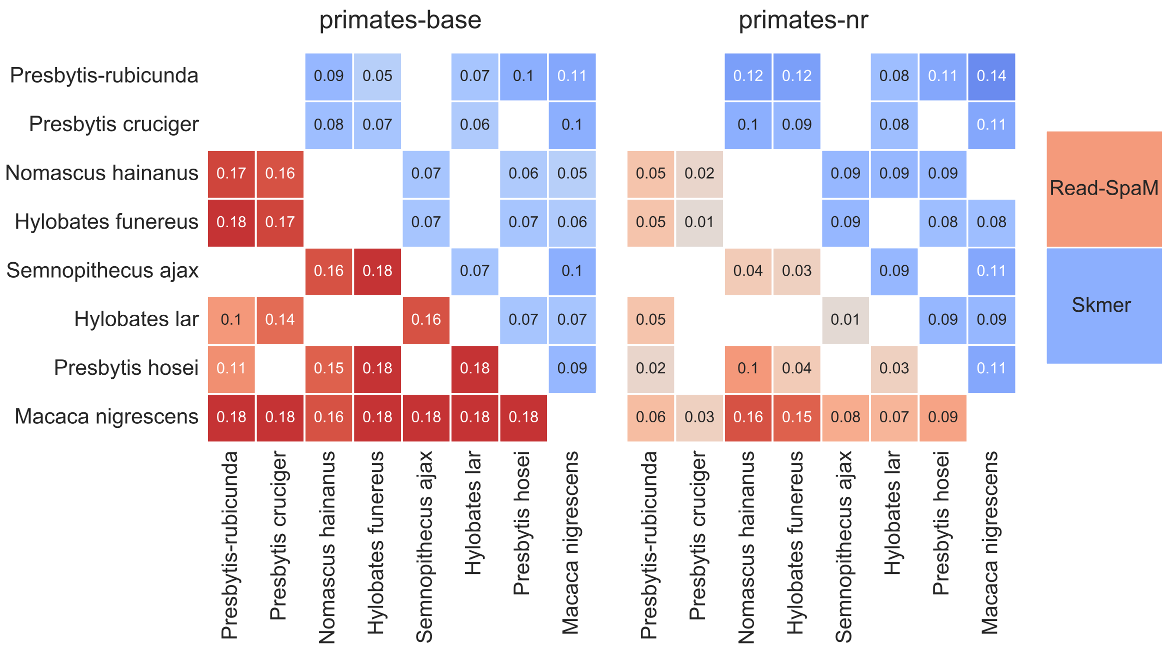


Figure 6.1 – Accuracy of distances calculated with READ-SPAM and SKMER. No exact reference distances are available for the eight species under consideration. Instead, we roughly estimated their sequence similarity from the most closely related species for which divergence times were given in literature. We refer to the estimates as d_e. The heat map on the left shows the difference $d_{\text{READ-SPAM}}(\text{PRIMATES-BASE}) - d_e$ (red) and $d_{\text{SKMER}}(\text{PRIMATES-BASE}) - d_e$ (blue). For certain pairs of species, no reference distances could be deduced (blank squares). The heat map on the right shows the difference $d_{\text{READ-SPAM}}(\text{PRIMATES-NR}) - d_e$ (red) and $d_{\text{SKMER}}(\text{PRIMATES-NR}) - d_e$ (blue).

undergone basic quality control, they still originate from both nuclear and mitochondrial DNA. In addition, they may also be contaminated by other microbial organisms. Therefore, we applied the program KAIJU [175] to remove all microbial DNA from the data set. KAIJU screens the reads against a large reference database of microbial sequences and assigns the appropriate taxonomic labels. The program provides several readily prepared reference databases; we used the collection NR+EUK which contains sequences from the domains of *Bacteria*, *Archaea*, *Viruses*, *Fungi*, and microbial eukaryotes. We refer to the resulting data set with exclusively nuclear and mitochondrial DNA from the target primates as PRIMATES-BASE. We separated mitochondrial reads from nuclear reads in PRIMATES-BASE by performing a local BLASTN search against a large selection of mitochondrial genomes sampled from NCBI. The two resulting data sets are termed PRIMATES-NUCLEAR and PRIMATES-MITO, respectively. Furthermore, we removed all repetitive regions from PRIMATES-BASE by applying REPEATMASKER [364] with default parameters; the data set without repeats is referred to as PRIMATES-NR. For each of the four data sets, we calculated pairwise distances between the eight samples with READ-SPAM and constructed a phylogenetic tree based on the resulting distance matrix using Neighbour-Joining. In addition to READ-SPAM, we also applied other alignment-free and assembly-free programs, namely AAF [443] and SKMER [444]. This allows us to compare the distance estimates of the programs with each other. Normally, it would be best to also compare the computed distances with robust reference distances, for example, inferred from whole genome alignments. However, no publicly available reference genomes

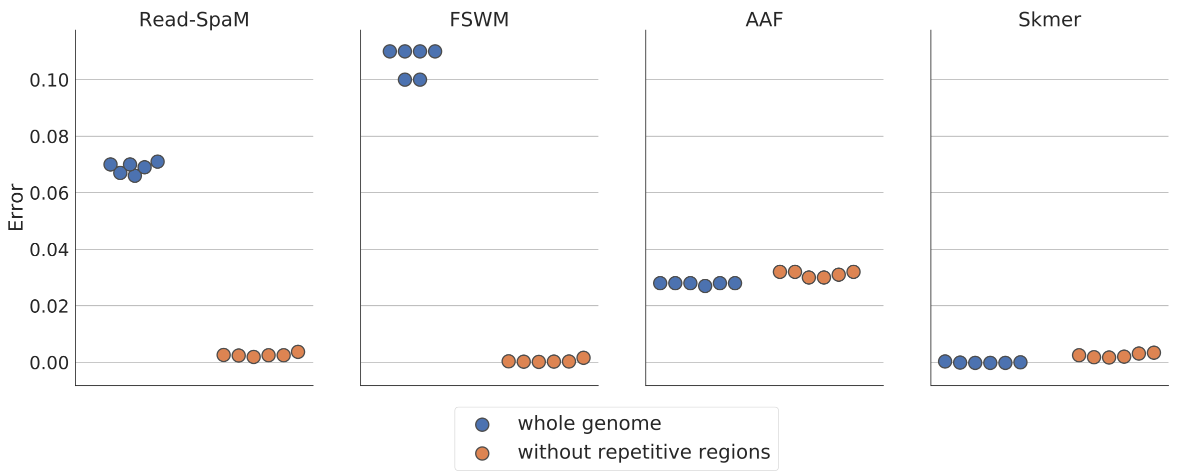
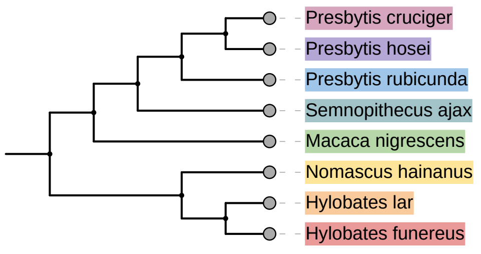


Figure 6.2 – Pairwise distance estimates of READ-SPAM, FSWM, SKMER, and AAF between four Great Apes—resulting in six distances each—of the UCSC data set. Distances were either calculated from the whole genomic data (blue dots) or from genomic data without repetitive regions (orange dots). The error (y-axes) is the absolute deviation of distance estimates from the respective alignment-free program compared to distances calculated from alignments.

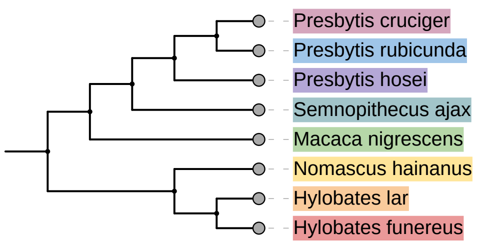
exist for the eight species under consideration. Instead, we prepared a data set, called UCSC, of four closely related species of Great Apes retrieved from the UCSC genome browser [445]. We calculated pairwise distances for this data set with READ-SPAM, FSWM, AAF, and SKMER. As a reference, we also computed distances by counting nucleotide mismatches of the corresponding whole genome alignments.

The analyses provided several insights: The first main finding is that READ-SPAM's distance estimates must be taken with caution on complex eukaryotic data sets that originate from whole genome shotgun sequencing. An overview of the precision of pairwise distances from READ-SPAM and SKMER is illustrated in Fig. 6.1. READ-SPAM severely overestimates distances on the PRIMATES-BASE data set, often up to 0.18 mutations per sequence position. Similarly, SKMER overestimates the distances, but with a magnitude that is half as large as that of READ-SPAM. When repetitive regions are removed from the genomes, the distance estimates of READ-SPAM improve significantly and their error drops up to 80% for many species; still, they remain rather inexact for several species pairs, especially those involving *Macaca nigrescens*. In contrast, SKMER is very accurate regardless of the presence of repetitive regions, but its distances get slightly worse when repetitive regions are removed. We confirmed these observations with the auxiliary UCSC data set of fully assembled genomes. The difference between the reference distances and the distances computed by the alignment-free programs is shown in Fig. 6.2. As before, READ-SPAM and FSMW strongly overestimate pairwise distances in the presence of repetitive regions, while distance estimates of SKMER are accurate. Without repetitive regions, READ-SPAM and FSWM produce very accurate distances, while SKMER gets slightly worse. The distance estimates of AAF are inaccurate throughout, regardless of whether or not there are repetitive regions.

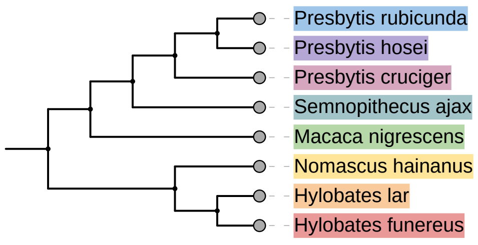
The second main insight is that the quality of inferred phylogenetic trees does not correlate with the absolute accuracy of distance estimates. Figure 6.3 presents four phylogenetic trees: The true known tree based on curated marker genes from mtDNA, and three trees inferred by READ-SPAM based on the data sets PRIMATES-BASE, PRIMATES-MITO, and PRIMATES-NR.



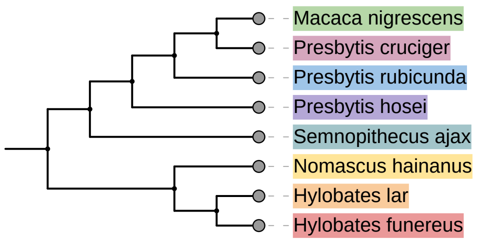
(a) True tree.



(b) READ-SPAM, PRIMATES-BASE.



(c) READ-SPAM, PRIMATES-MITO.



(d) READ-SPAM, PRIMATES-NR.

Figure 6.3 – Phylogenetic trees for the eight Old World monkeys inferred with READ-SPAM from different data sets, as indicated. The first three trees (a) to (c) differ only with respect to the clade of three *Presbytis* species. The last tree (d) is inferred from repetitive regions and varies with respect to the location of *Macaca nigrescens*.

All trees are presented as cladograms without branch lengths. The only difference between the topology of the first three trees is the arrangement of three distal branches that comprise closely related species of the genus *Presbytis*. The topology of the PRIMATES-BASE tree is also identical to topologies based on distances from SKMER and AAF; thus, it appears that this topology is robust when all nuclear data is taken into account. In contrast, the topology of the last tree, which is based on the data set PRIMATES-NR, deviates from the other trees with respect to the position of *Macaca nigrescens*. This species is incorrectly placed within the clade of the other *Presbytis* species. However, for this data set, the absolute error in the distance estimates was the lowest out of all data sets. This constitutes a discrepancy between the precision of distance estimates and tree topologies: the only data set where the error of estimated distance decreases—namely PRIMATES-NR—is also the only data set with a strongly divergent tree.

These observations imply several conclusions: First, methods based on spaced-word matches struggle to deal with repetitive regions. We infer this from the strong improvement of distance estimates when repetitive regions are removed in any data set compared to including repetitive regions. This observation is most likely explained by the very nature of our method: Estimating the number of nucleotide substitutions per sequence position precisely requires that all spaced-word matches from non-homologous sequence positions are excluded. With our current filtering procedure, it is highly unlikely that this precondition holds: The presence of repetitive regions causes many spaced-word matches from non-homologous sequence regions with unusually high similarity. Although these matches pass the filtering threshold, they share fewer similarities than true homologous regions. We confirmed this theory by investigating spamograms of varying data sets. We observed that a distinct homologous peak was missing in the presence of repetitive regions; instead, a wide and flat distribution of scores above zero occurred with a mean score well below what is expected. On the contrary, a distinct homologous peak reappeared when repetitive regions were removed, see Suppl. Fig. D.1.

Second, repetitive regions are not the only artifact that leads to inaccurate distance estimates in PRIMATES-BASE. We come to this conclusion because READ-SPAM and SKMER are very accurate on the UCSC reference data set, but not on the PRIMATES-BASE data set. Furthermore, the number of identified repetitive regions varies strongly between the eight species and is the lowest for *Macaca nigrescens*: For this species, the distance estimates remained largely incorrect (see Fig. 6.1) and its position within the inferred trees was wrong. We assume that its corresponding sample contains sequencing errors or biases that we did not catch in our preprocessing steps. Such artifacts might originate from the sequencing itself, from unidentified repetitive regions, from particularly low or biased sequence coverage, or from other unidentified organisms that were not detected by KAIJU.

Third, the tree topology recovered from the complete nuclear DNA of the samples indicates that the three species of the genus *Presbytis* might have hybridized. They all occur in the same geographical area on the island of Borneo and their hybridization is possible [pers. comm. with C. Roos (German Primate Center), April 2021]. This theory is promoted by our findings that the tree inferred from mitochondrial DNA deviates from all trees inferred from nuclear DNA, regardless of which alignment-free program was employed.

Although the results of this study remain inconclusive with respect to the accuracy of distance estimates, the quality of the available ancient DNA samples, and the true reason for the observed differences in tree topologies, our analyses brought valuable insights for the application of SpaM-based methods on eukaryotic data sets. These insights will shape future work on the development and application of SpaM methods, especially in regard to considering eukaryotic sequence data. Furthermore, we also noticed inconsistencies in the distance estimates of both other alignment-free methods, which demands a more detailed analysis. In general, alignment-free methods should be continuously integrated into studies that focus on more complex real-world data as also postulated by efforts such as AFproject [344].

6.2 Using Simon's Congruence to Estimate Sequence Similarity

A plethora of methods are available that estimate the similarity of two or more DNA or protein sequences. We discussed alignment-based approaches in Sec. 2.3 and alignment-free approaches in Sec. 2.4. A large subgroup within the latter are word-based methods which characterize the input sequences by transforming them to new feature spaces: The most common approach is the use of *contiguous* substrings of a *fixed* length, also referred to as words or k-mers. The similarity between sequences can then, for example, be estimated from the presence, absence, or abundance profiles of the k-mer sets present in the sequences [446, 447]; even the succession of multiple k-mers has been examined in this context [448]. Specialized programs have been developed that extract and save k-mers with high speed and low memory demands [449, 450]. In some approaches, the length of the words is not fixed but may vary, for example, when determining shortest unique substrings [154] or longest common substrings [451] between sequences. Furthermore, *non-contiguous* words of a fixed length have been employed to assess sequence similarity; here, one example are the SpaM approaches [452] which are discussed in detail in Subsec. 2.4.3. All of the above methods rely either on non-contiguous or contiguous words of a *fixed* length or on contiguous words of a variable length. To the best of our knowledge, no approach exists that employs *non-contiguous* words of a *variable* length to measure the similarity between molecular sequences.

As defined in Sec. 2.2, for a sequence S with $|S| = n$, a subsequence of length $k <= n$ is a non-contiguous string of symbols $S[i_1]S[i_2] \dots S[i_k]$ defined by k indices $i_1, \dots, i_k$ with $i_q < i_r$ for $q < r$, $q, r \in [k]$. In contrast to substrings, the use of subsequences has found little interest in computational bioinformatics (note that several authors use the term subsequence to refer to a contiguous substring). One reason for this could be the increase in computational complexity when considering subsequences. While the number of substrings of a fixed length k grows linearly with sequence length n, the potential number of subsequences of fixed length k increases as a polynomial function of order k with increasing n: the number of subsequences is given as $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. Thus, it is unfeasible to use *all* subsequences of a molecular sequence for any real world use case where genes, scaffolds, or genomes have to be compared. Instead, we examine one specific measure between two strings that is based on subsequences, namely, the Simon's congruence: For a sequence S, let $\mathcal{S}^k$ be the set of all distinct subsequences of S with length k. Two sequences S_1 and S_2 are Simon-k-congruent—also written as the equivalence $S_1 \sim_k S_2$—if $\mathcal{S}_1^k = \mathcal{S}_2^k$. In other words, the set of subsequences of length k are identical in S_1 and S_2. The problem of finding the largest $k \in \mathbb{N}$ for which $S_1 \sim_k S_2$ holds is known as MaxSimK. Recently, the first algorithm to solve MaxSimK in linear time with respect to the sum of the lengths of the input sequences has been published [453] and an implementation of this algorithm has been made available to us. While the Simon's congruence has applications in learning theory and linguistics [453], it has never been used for biological sequence comparison as far as we are aware. On the most fundamental level, we explore whether MaxSimK carries any signal to deduce the similarity of biological sequences.

For this, we designed multiple simple experiments to explore the behaviour of MaxSimK with respect to sequence length, sequence similarity, and further biological evolutionary effects such as indels. To have maximal control over the sequence properties, we created sequences artificially using a straightforward simulation framework: We simulate novel sequences over the alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ using a uniform distribution over the nucleotides and independent sequence positions. For a simulated sequence S of length n, we add a mutation by choosing a sequence position $i \in [n]$ uniformly at random and exchanging the nucleotide at $S[i]$ by *another* nucleotide. For a specified mutation rate r, we perform such an exchange repeatedly for exactly $r \cdot n$ times. We simulate an insertion or deletion (indel) by choosing a sequence position $i \in [n]$ uniformly at random. Next, we decide whether to simulate an insertion or deletion at random; although deletions are more common in some real-world data [416], we assume that insertions and deletions occur with equal probability. This ensures that the expected sequence length remains n on average when adding multiple indels. We determine the length n_i of the indel by drawing a random sample from the Zipf power law distribution with parameter $a = 1.7$. However, we limit the maximal length of an indel to 10 bp. For a deletion, we remove the sequence elements $S[i : i + n_i]$; if $i + n_i$ is larger than n, we delete the last n_i elements of S instead. For an insertion, we insert a newly simulated sequence of length n_i into S, starting after nucleotide $S[i]$. For a specified indel rate r, we perform this operation repeatedly for exactly $r \cdot n$ times.

First, we tested how MaxSimK behaves for two sequences of the same length n. We simulated ten DNA sequences of fixed length $n = 500$ and calculated MaxSimK for each sequence pair, resulting in 100 measurements. The procedure was repeated for further lengths $n \in \{1\,000, 1\,500, \dots, 4\,000\}$. Figure 6.4 displays the average value of MaxSimK of all 100 simulated sequences of identical length, dependent on the sequence length. There is a distinct linear relationship between the length of the involved sequences and MaxSimK. The variance of MaxSimK across all sequence pairs of the same length is minimal (note that the standard deviations in Fig. 6.4 are multiplied by a factor of 50). A linear regression based on all

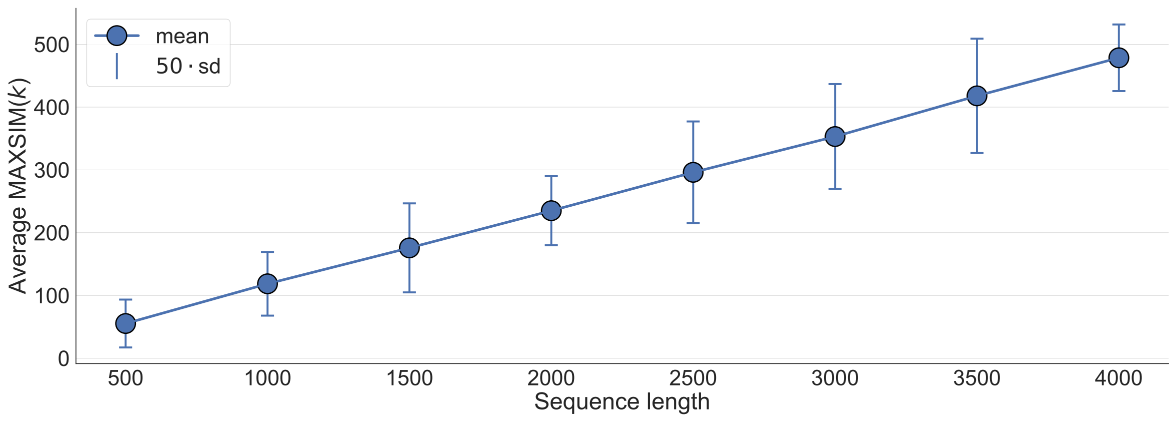


Figure 6.4 – Average MaxSimK between sequences of length n with respect to n. The sequence length n is increased from 500 bp to 4,000 bp in steps of 500. For each value of n, the average MaxSimK of pairwise comparisons between ten sequences is shown (blue dots). Error bars indicate 50 times the standard deviation.

pairwise values yields the relationship

$$\text{MaxSimK}(n) = 0.1203 \cdot n - 4.372 \tag{6.1}$$

that describes the expected value of MaxSimK for two sequences of length n; the linear relationship is well supported ($r^2 = 0.9998$).

Second, we evaluated how MaxSimK behaves for two input sequences of different lengths. For this, we simulated five DNA sequences of length $n \in \{500, 1\,000, \ldots, 5\,000\}$, respectively. Then, we calculated MaxSimK between each sequence pair regardless of their length. Figure 6.5 shows MaxSimK between sequence pairs when the length n_1 of *one* of the two sequences is fixed while the length of the other sequence varies. These experiments are further broken down in Fig. 6.6, which shows pairwise MaxSimK values for each combination of sequence lengths separately. The figures reveal that MaxSimK depends almost exclusively on the length of the shorter sequence and is very robust therein. For example, all pairwise comparisons involving the five simulated sequences with length $n_1 = 5\,000$ closely follow the linear relationship specified in Eq. 6.1 with respect to the length of the second sequence.

Third, we simulated 20 base sequences of length $n = 500$. For each sequence, we created nine mutated copies for mutation ratios of $r \in \{0.1, 0.2, \ldots, 0.9\}$, resulting in a total of 200 sequences. For each of the base sequences, we calculated MaxSimK to each of its mutated versions. Figure 6.7 shows all resulting MaxSimK values with respect to the respective mutation rates. Equally, Suppl. Fig. E.1 presents results for sequences of length $n = 1\,000$. On average, MaxSimK stays near constant across the mutation rates. A Welch t-test between any two mutation rates is non-significant ($P > 0.05$) for both $n = 500$ and $n = 1\,000$. As before, MaxSimK solely depends on the length of the sequences according to Eq. 6.1.

Fourth, we repeated the above experiments, but this time with added insertions and deletions (indels) of a specified rate in addition to mutations. Again, we used sequences with a length of $n = 500$ while the mutation and indel rate was either kept low with values of $r \in \{0.01, 0.02, \ldots, 0.09\}$ or $r \in \{0.1, 0.2, \ldots, 0.9\}$. The incorporation of indels barely influences the results, see Suppl. Fig. E.2 and Suppl. Fig. E.3. Again, there is no statistically significant trend of the average MaxSimK with respect to the mutation and indel rate. We also altered the experimental setup by simulating all mutated sequences from the same base

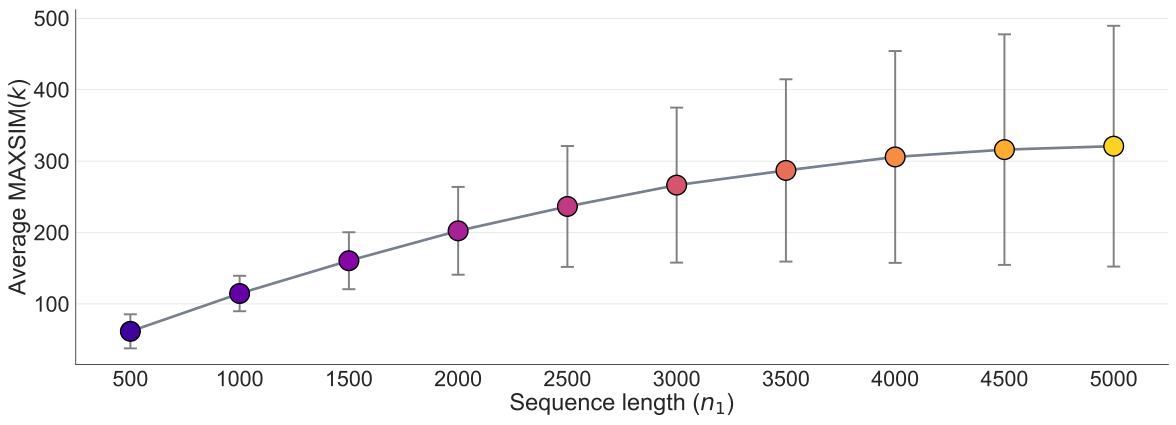


Figure 6.5 – MAXSIMK between sequences of different length. For all sequences of length n_1, the average MAXSIMK to all other sequences of any length $n_2 \in \{500, 1\,000, \ldots, 5\,000\}$ are shown.

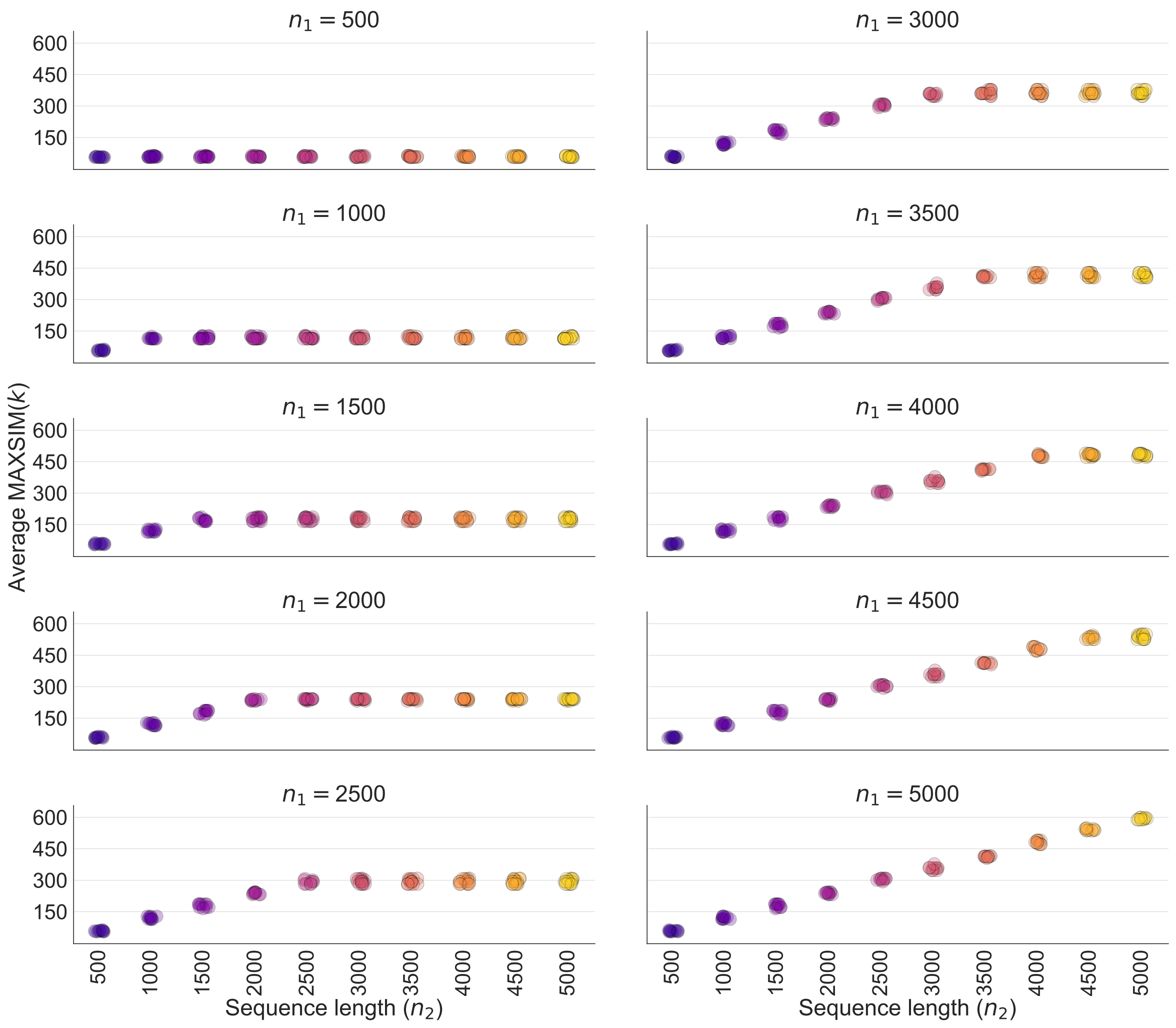


Figure 6.6 – MAXSIMK between sequences of different length. For each sequence of length n_1, the MAXSIMK to all other sequences of any length n_2 are shown (individual plots). Each dot represents the MAXSIMK value for exactly one pair of sequences.

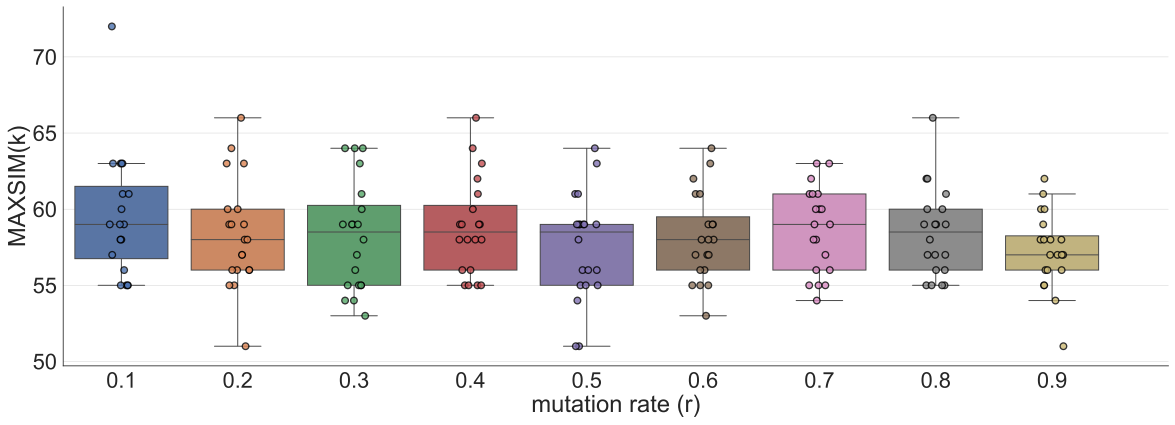


Figure 6.7 – MaxSimK between sequences of length 500 with different hamming distances. We calculated MaxSimK between sequence pairs for which mutations with a rate r were simulated. Mutation rates were varied from 0.1 up to 0.9 average substitutions per sequence position.

sequence and calculating MaxSimK between all pairs of sequences with the same mutation and indel rate. The according results for low rates r are presented in Fig. 6.8, and for high rates in Suppl. Fig. E.4. In contrast to previous experiments, here we observe an unambiguous increase in variance for larger mutation rates. Between any two mutation rates, the variance is statistically significant as assessed by Levene's test [368] with a significance level of $\alpha = 0.05$. The increase in variance most likely originates from the simultaneous increase in variance of the sequence length that is caused by higher indel rates in the experimental setup.

Fifth, we calculated MaxSimK for all sequence pairs of a real-world data set comprising 150 16S genes. The results are visualized as a heat map in Fig. 6.9. In accordance with our previous observations, MaxSimK is always driven by sequences that possess unusual subsequences and thus, result in small MaxSimK values. The heat map reveals an according

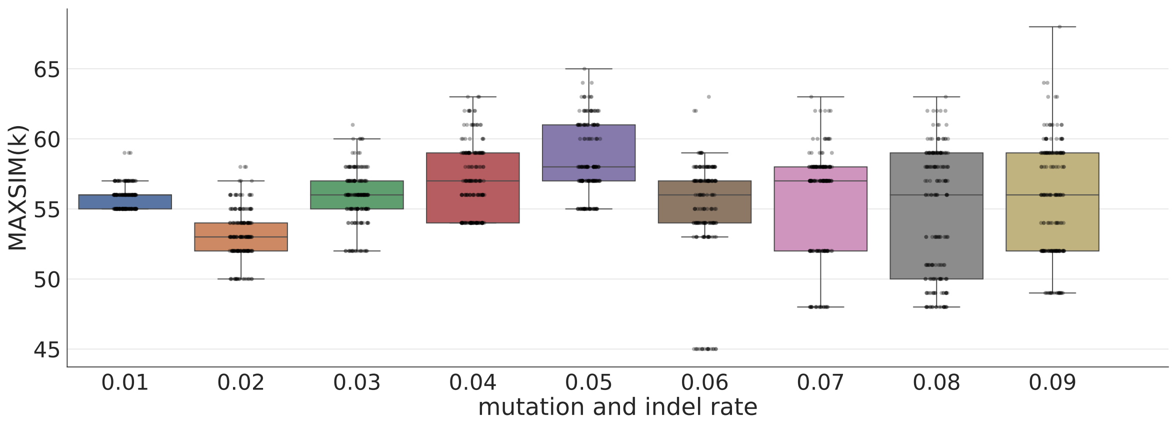


Figure 6.8 – MaxSimK between sequences of length 500 with simulated indels. MaxSimK was calculated between all sequence pairs for which mutations and indels were simulated with the same rate r (black dots). Mutation rates were varied from 0.01 up to 0.09 average substitutions per sequence position. Average statistics are over all sequence pairs of the same mutation rate are shown (box plots).

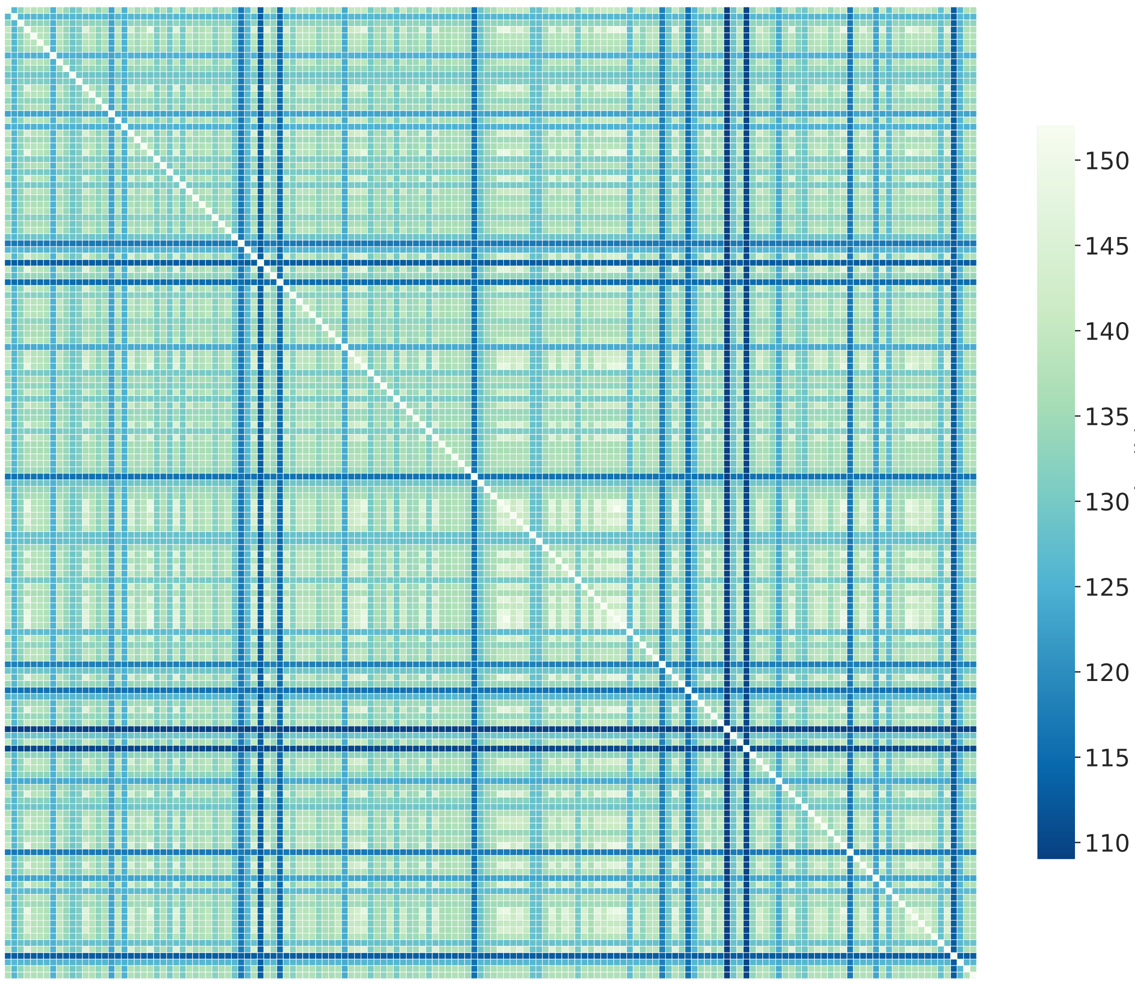


Figure 6.9 – MaxSimK between sequence pairs of bacterial 16S sequences. Each colored square represents MaxSimK for a single sequence pair. The data set comprises 150 bacterial sequences, resulting in 22 500 pairwise comparisons. MaxSimK varies from 109 to 152.

'checkerboard' pattern where low MaxSimK values are dominant for the respective sequences. We converted the MaxSimK values to a distance matrix and applied NJ to reconstruct a phylogenetic tree. The resulting tree bears little resemblance to reference trees constructed from multiple sequence alignments and exhibits a nearly maximal RF distance to those trees.

The presented results suggest that the Simon's congruence does not provide helpful information to assess the similarity of DNA sequences. Instead, our experiments demonstrate that MaxSimK has no correlation with the degree of relatedness of the input sequences, see Fig. 6.7. This applies to sequences of different Hamming distances, as well as to sequences with simulated indels. Instead, MaxSimK is solely dependent on the length of the shorter of the two input sequences where longer sequences entail higher values of MaxSimK. The algorithm calculates MaxSimK in linear time with respect to the sum of the lengths $n_1 + n_2$ of the two input sequences. We were supplied the very first implementation of the algorithm which is written in the programming language Python. Although the algorithm theoretically runs in $\mathcal{O}(n_1 + n_2)$, the provided implementation has limited computation speed due to high constant operational costs. If MaxSimK were to be applied for large data sets or for sequences spanning large regions of a genome, an updated implementation would be necessary. The algorithm does not only supply the answer to MaxSimK but may also output any sequence of length $k+1$ which is present in one of the two sequences and not the other. We did not utilize this information so far, but including it in future work might create novel opportunities.

Our evaluation pipeline was deliberately kept simple and did not use sophisticated sequence simulation frameworks to reduce any effects that may stem from the simulation itself. While this provides clear results, it also limits the extent to which our results can be generalized. For example, all sequences were simulated with a uniform distribution over Σ; however, real sequences often exhibit regions which are not uniformly distributed; for example, a GC bias is present in most sequence regions. Any such irregularity will also affect MaxSimK, but the extent to which this happens requires further examination. One idea to overcome the observed limitations was to group nucleotide sequences into consecutive k-mers and consider those k-mer sequences instead of the original ones. We tested this approach for $k = 2$ by translating input DNA sequences into another alphabet Σ' with $|\Sigma'| = 16$ and ran MaxSimK on the altered sequences. However, increasing the size of the alphabet causes a strong reduction in MaxSimK values. Often, already two or three consecutive symbols of Σ' occur exclusively in a single sequence, limiting MaxSimK values between any two sequences to 2 or 3, see Suppl. Fig. E.5. Further experiments could consider the minimal diverging subsequence that is output by the algorithm or refrain from the use of the Simon's congruence altogether.

Chapter 7

Discussion

Exploring the relationships of Earth's enormous diversity of living organisms based on molecular sequence data has been at the heart of biological sciences for nearly 50 years [454]. Characterizing similarities and differences between organisms has a long history, from analyzing morphological features to individual genes, to whole genomic, transcriptomic, and proteomic data. Thereby, the field of phylogenetics—as well as phylogenomics—emerged: the study of the evolutionary relationships of species by investigating their DNA, RNA, or protein sequence data. Thus, there is a steady and continuous change of the very thing that is at the core of phylogenetic analyses. This observation also yields the insight that it is not about observing and modeling processes of nature, but about *how* nature is observed and modelled. Each model represents only one of many approaches to generate a consistent framework and may become inconsistent or outdated at any time. By analyzing how the model came to be, how it is utilized, and how it holds up to current research, we may learn about its virtues and flaws. In this spirit, I attempt to not only summarize what we observed, but also how we observed and how this may guide future research to create improved models.

Phylogenetic inference is at the basis of all analyses presented here. The universal goal is the reconstruction of a phylogenetic tree that showcases the evolutionary relationships of the involved entities; in the biological sciences, these entities are usually species or genes. However, phylogenetic analyses have also been used, for example, in linguistics [455] or for the examination of the history of law [456]. The sentiment of representing the evolutionary history of species in a tree-like manner has been popular since Charles Darwin proposed a tree-like structure instead of independent lineages to interrelate a group of finches [457]. The nodes in a phylogenetic tree represent ancient species for which a speciation event occurred, while the branches depict evolutionary alterations of the species. Although it is well known by now that biological processes do not necessarily behave in a tree-like manner, phylogenetic trees remain the most common form of representing evolutionary relationships up until today. Two examples of processes that conflict with the assumption of tree-like evolution are horizontal gene transfers and hybridization events. In the former, one or multiple genes are transferred from one organism to another, resulting in a genome composition that does not follow a single evolutionary path; instead the genome may be comprised of segments from multiple other organisms. Similarly, hybridization is the fusion of species and cannot be represented by a tree structure: a hybridization violates the acyclic nature of a tree and can be seen as the opposite to a speciation event. Therefore, the adequacy of using *networks* instead of trees to model and visualize phylogenetic relationships has been extensively discussed [458, 459]. However, there is substantial uncertainty as to what extent horizontal gene transfer impacts phylogenetic relationships. After all, the prevalence of horizontal gene transfer varies considerably between different phyla. Although there are prominent examples of lasting HGT events [460, 461], it has been argued that the influence of HGTs in the large picture of evolution has been overstated and that HGT events rarely consolidate in a species population [462]. HGT may still be prevalent in certain taxonomic groups, such as in bacterial evolution; it is also known that HGT occurs regularly in the mitochondrial genomes of plants, although the mechanisms by which these transfers occur are not yet clear [463]. In contrast, HGT in plants occurs rarely in the chloroplast and nuclear genomes [463].

In general, phylogenetic trees are a subtype of phylogenetic networks, whereas the structural constraints of trees prevent their applicability to all evolutionary relations. Although these considerations suggest that phylogenetic networks are superior to phylogenetic trees, the latter also offer advantages: Phylogenetic trees are easier to represent and comprehend. They also have a long and extensive history of usage, which resulted in sophisticated and versatile tree reconstruction algorithms and a broad assortment of additional frameworks that center and operate on phylogenetic trees. This environment facilitates the further use of trees in contemporary research, even in those cases where networks may be more appropriate. For similar reasons, we exclusively used tree-like representations of phylogenetic relationships, undoubtedly an inherent limitation of our work. Nevertheless, the use of phylogenetic trees remains reasonable in many real-world scenarios that are concerned about the evolutionary development of species where reticulate evolution is infrequent. In addition, the use of alignment-free methods in general is not limited to phylogenetic trees, but the resulting distance matrices can also be used to infer networks with appropriate software [464, 465]. However, studies that assess the precision and biological significance of the resulting networks are limited.

For a long time, the reconstruction of phylogenies based on molecular data has mainly relied on sequence alignments and maximum likelihood (ML) estimation based on DNA, RNA, or protein sequences [466]. Other common approaches to infer phylogenies include Bayesian estimation, alignment-free methods, and maximum parsimony, all of which come with advantages and drawbacks. ML and Bayesian methods are among the most precise, while alignment-free distance-based methods compensate for lower accuracy with reduced time and memory requirements; thus, the latter can be used on large data sets where ML methods are not applicable. Especially the emergence of second- and third-generation sequencing techniques that produce large amounts of data made it necessary to devise more efficient methods. Another drawback of both ML and Bayesian methods is their reliance on sophisticated evolutionary models. Specifying adequate model parameters can be an additional source of error, and the choice of an optimal model and its parameters remains an ongoing debate in phylogenetics [234, 467]. On the contrary, Maximum Parsimony (MP) does not assume an underlying model and performs best when the number of evolutionary changes remains low [468]. The spaced-word matches (SpaM) approaches are a group of distance-based methods that tackle tasks within phylogenetics and metagenomics. In contrast to many other alignment-free word-based methods, SpaM approaches do not use contiguous words. Instead, they find segments in the input sequences that share identical nucleotides with respect to a predefined binary pattern. In doing so, they estimate the evolutionary distance between the input sequences by calculating the average number of nucleotide substitutions per sequence position. The term *alignment-free* may be misguided: Although SpaM methods do not rely on local or global alignments in the traditional sense, pattern-based matches can be seen as a large collection of *micro-alignments*.

Despite algorithmic advances, phylogenomic methods barely keep up with processing the large volume of molecular data produced on a daily basis. A substantial amount of the produced data consists of short sequencing reads of unknown origin, and their taxonomic identification—the read assignment—is often one of the first steps in subsequent processing pipelines. A promising approach that blends the areas of phylogenetics, phylogenomics, and metagenomic read assignment is *phylogenetic placement* (PP). In PP, molecular sequences are placed onto an existing reference phylogeny to characterize their phylogenetic relationships with the reference sequences of the provided tree. By its very design, PP aims to circumvent some of the basic problems that subsist in phylogenetics: Not only does it require a fixed phylogenetic tree as input, but it furthermore assumes that this tree has been created with state-of-the-art standards and represents the underlying evolutionary relationships between the species as best as possible. Here, App-SpaM has been presented, a novel alignment-free approach to PP that is based on spaced-word matches and provides intriguing and novel applications for PP.

7.1 Evaluating Phylogenetic Placement Algorithms

Phylogenetic placement presents a unique mechanism to observe the evolutionary relationships of sequences: By placing query sequences directly into an existing phylogenetic tree, the time-intensive step of de-novo tree reconstruction is omitted; this allows the phylogenetic characterization of a large number of reads based on high-quality reference phylogenies. Several algorithms for PP have been developed in recent years that pursue different strategies to derive suitable placement locations. Although the first approaches were based on maximum likelihood calculations on the basis of multiple sequence alignments [10, 12, 15, 330], more recent methods pursue alignment-free or mixed strategies [13, 14, 177, 332]. Due to the computational demands of ML-based methods, PP has been applied mainly for amplicon-based sequencing studies where reference and query sequences originate solely from designated short genomic regions. Furthermore, such types of sequencing data allow the straightforward construction of multiple sequence alignments. We proposed the alignment-free PP tool App-SpaM, which performs placement based on filtered spaced-word matches, a versatile technique that allows App-SpaM to function on both assembled or unassembled DNA or RNA. Consequently, it is possible to apply App-SpaM on a wide variety of data sets where PP could not be applied previously. The only other tool that currently works on unassembled data is APPLES, however, it also infers phylogenetic distances from alignments in its default version.

Like for alignment-free phylogeny reconstruction, the typical trade-off between speed and accuracy also applies for phylogenetic placement. Which PP algorithm is the 'best' depends on the type of input sequences, the available computational resources, the proficiency of the user, and the requirements on the quality of resulting placements. The large variety of PP approaches demands a comprehensive evaluation; not only to illustrate their advantages and drawbacks, but also to guide end users to an informed decision about which tool to use with respect to their data and research question at hand. However, no universal evaluation of all available methods has been conducted so far. Instead, each software performed separate benchmarks following various workflows and employing different metrics to measure placement accuracy; see also Subsec. 2.7.3. We identified three distinct factors which are responsible for this: First, there was no single established evaluation procedure that has been generally accepted by the research community. Second, different accuracy metrics are available and have been used in different works. Third, several approaches have been developed and published concurrently in recent years, making it challenging to compare them among each other. Fortunately, the first two issues were recently addressed by the Placement Evaluation Workflows (PEWO). PEWO offers a common foundation for the analysis of PP tools and proposes reproducible workflows and clearly defined metrics. Internally, PEWO is based on the Snakemake framework [348], which allows an easy extension of PEWO as well as its integration into other software packages.

Based on PEWO, we performed a comprehensive evaluation of all readily available placement programs with which we compared our novel approach App-SpaM. One strength of PEWO is the unification all placement programs in a single Conda environment that can be installed effortlessly by end users. We ran PEWO's pruning-based accuracy evaluation (PAC) workflow on a broad array of metataxonomic data sets compiled from different sources. Our evaluation revealed that App-SpaM is on par with alignment-based programs under the ND metric on a wide variety of metataxonomic data sets, although App-SpaM only applies rough heuristics to infer placement positions. Furthermore, its runtime is two to three orders of magnitude faster than that of the next fastest PP programs. The results are consistent for a large number of parameter configurations of App-SpaM, especially with respect to the number and structure of the binary patterns that are employed. An in-depth analysis of those results is given in Subsec. 3.1.2 and we refer to Subsec. 3.1.3 for a detailed discussion of the advantages and disadvantages of our algorithmic approach. In Sec. 4.3, we proposed a sampling procedure for App-SpaM that greatly accelerates its placement speed and is especially useful when large collections of unassembled sequencing reads are available as reference sequences. In addition, we suggested a simple measure to specify the placement uncertainty of App-SpaM, see Sec. 4.4. By this, it would be possible to filter placements with respect to their reliability; unfortunately, our presented method did not produce reasonable results so far. Still, we believe that the novel design of App-SpaM and our associated developments are an important contribution to promote and diversify the field of phylogenetic placement algorithms. App-SpaM does not only handle novel use cases for PP, but also raises questions about the current state of PP tools, frameworks, and their evaluation in general.

PEWO constitutes an important milestone in the development and progression of phylogenetic placement algorithms and efforts must be made to promote its role within the community. Nonetheless, PEWO also has shortcomings that should be addressed to ensure its longevity: The PAC workflow of PEWO delineates the main approach to determine the quality of placement programs. Although we agree that the underlying PAC procedure constitutes a reasonable basis to assess placement accuracy, it is not versatile enough in its current state to handle the variety of disparate input and output data that may be subject to phylogenetic placement. Currently, the workflow is limited to typical metataxonomic data sets and requires an alignment of the reference sequences. Performing placement on unaligned references is not possible, even if the algorithms under consideration do not require aligned sequences. Furthermore, we have argued that there is no compelling need for the reference sequences to be assembled, see Subsec. 3.1.3; instead, PEWO should allow the use of standardized file formats for assembled and unassembled reference sequences as long as subsequent placement programs can handle such data. Moreover, query sequences are simply created by splitting the pruned reference sequences into fragments of equal length. The length of the query sequences is a major driving factor for the accuracy of placement programs, as shown in Fig. 3.10 and discussed further for the placement of complete genes in Subsec. 5.1.3. In addition to the query length, the possibility to simulate query reads of arbitrary coverage and with varying degrees of sequencing errors should be added, since the impact of such alterations remains unclear; see, for example, Fig. 3.11. In addition, PEWO only implements two metrics to describe placement accuracy; namely, the node distance (ND) and expected node distance (eND). Although these metrics are useful for gaining basic insights into the accuracy of placement programs, they only capture one aspect of the multifaceted placement quality. The metrics ND and eND are both conditioned on the total number of leaves present in the reference tree; a *normalized* ND metric that is independent of the size and structure of the reference tree would allow the comparison of placement accuracies between data sets. More specifically, as introduced in Subsec. 3.1.1, *control methods* are a powerful tool to assess how much a placement program improves upon random or simplistic placement methods. Control methods can be as simple as placing each query at the root, at the midpoint, or at the topological midpoint of $\mathcal{T}_{\text{ref}}$ (whereas the topological midpoint is the branch that has the lowest average node distance to every leaf). Considering the normalized difference in placement accuracies between PP methods and control methods would be a powerful tool to establish an independent accuracy measure.

Additional metrics, besides those that utilize a—potentially normalized—ND metric, might comprise the delta metric for simulated data sets where the underlying real tree is known, or the KR distance between the placements of different program runs, see Subsec. 2.7.3. The latter would be an efficient manner to quantify pairwise differences between all programs under consideration in a single program run. All of the above metrics neither take into account the topology of the reference tree, nor do they incorporate the inferred branch lengths for the pendant, proximal, and distal branches. A straightforward extension of the ND metric would be to calculate the difference in branch lengths between the proposed and expected placement locations instead of the number of nodes. We would expect that non-likelihood-based programs such as App-SpaM and RAPPAS may perform slightly worse when branch distances are incorporated as they do not rely on sophisticated evolutionary models. However, even certain ML-based methods such as EPA-ng merely use predefined default values for the newly added branches by default in order to speed up computation time. When exact branch lengths are calculated, their computational needs increase even further. Dedicated tests are required to appropriately evaluate these speculations.

Although there is a variety of potential input file formats for PP programs, their output is standardized by the JPlace format [329]. Since JPlace builds on the versatile json format [469], JPlace files are flexible and can, in theory, accommodate information about any aspect of the placement process and placement locations. However, the originally proposed version of JPlace suggests only five distinct values for each query, indicating the placement branch, the likelihood score, the likelihood-weight-ratio, the pendant branch length, and the distal branch length. Additionally, the same placement information can be assigned to multiple query sequences, and a multiplicity may be encoded for query sequences, for example, to retain count values from operational taxonomic units. This implementation of the JPlace format is now commonly integrated in associated software programs. Any additional information within the placement file is poorly understood at best, or brakes JPlace parsers and associated software tools at worst. Due to the recent diversification of use cases for PP methods, a universally accepted update of the JPlace format that retains additional information is vital. Such information should comprise general remarks on the types of input sequences—for example, whether reference sequences are assembled or aligned—or the method by which the reference tree was deduced. Query-specific information may contain details about the origin of samples, sample identifiers, and other associated information. Furthermore, additional query data should contain word count statistics for alignment-free programs. An extension of JPlace files has been suggested previously [334]. Despite these deficiencies of PEWO, we encourage the community to utilize this precious resource. A common framework is valuable, and its extension by the community is indispensable to overcome its deficits. Unfortunately, some recent work does not take advantage of this opportunity and instead uses separate evaluation procedures based on the delta error [332]. Although there may be convincing arguments to use other error metrics than ND and eND, and although there may be justified criticism on aspects of the PEWO framework, the evaluation of PP programs with non-publicly accessible evaluation software should be discouraged. It would be far better to extend the PEWO framework as a community effort to arrive at a common evaluation procedure.

While we discussed the potentials and shortcomings of evaluation methods in detail, we skipped one of the most important factors that influences evaluation efforts: The underlying data sets that are used. The choice of data sets is tightly coupled with other properties of the evaluation procedure, especially with respect to four key issues: First, for certain metrics—such as the delta error—the use of purely simulated data sets is inherently mandatory. Second, depending on the PP programs involved in the evaluation, the input data must comply to certain requirements; for example, it may need to be aligned or of a confined size due to runtime issues. Third, the data sets should not have been used for the evaluation of any of the PP programs itself, an issue that we ourselves violated as discussed in Subsec. 3.1.3. Fourth, the reference tree should be of high quality and suited for the placement task and query sequences at hand. We believe that an evaluation that is performed purely on synthetic data sets is hardly meaningful when its results are transferred to real-world data. Consequently, the exclusive use of the delta error might underestimate the error when placement is performed on real-world data sets. In this respect, there is additional work to be done for our evaluation on the continuous augmentation of phylogenetic trees and the detection of gene and species outliers. A comprehensive evaluation for a specified area of application should contain a variety of data sets from mixed sources. We hope that a collection of data sets for each area of application will crystallize from the recent efforts on phylogenetic placement. Furthermore, such data sets should be made publicly available, comparable to existing collections of data sets for wide-ranging tasks in alignment-free programs [344].

7.2 Applications of Phylogenetic Placement

Given an appropriate selection of reference sequences for the query sequences under consideration, phylogenetic placement is a valuable tool in phylogenetics and metagenomics: The major application of phylogenetic placement has been the identification of short metataxonomic read sequences [470–472]. Here, placement results provide information on the identity and evolutionary context of short sequencing reads. In this sense, PP is comparable to taxonomic read assignment with the difference that evolutionary relationships are directly estimated. In contrast, the use of taxonomic reference databases for read assignment can pose difficulties because common taxonomies differ from each other [473]; merging taxonomic and phylogenetic information into a singular tree has been attempted, but proves to be challenging [474]. After placing short reads, the results are commonly used to answer questions about the abundance of organisms in different environments, about the functional profiles of metagenomes, or about the inter- or between-sample diversity [382]. In metagenomics, PP has also served as a means to cluster sequences into operational taxonomic units [475]. There exists a comprehensive review on metagenomic data analysis using PP in general [334]. In addition to metagenomics, PP has also been applied for the approximation of the quality and completeness of microbial genomes [338], or for the identification of ancient DNA sequences [476]. As discussed in much detail in Sec. 5.1, three recent methods perform the iterative augmentation of existing phylogenetic trees by means of PP. Most of the mentioned programs have been published very recently, highlighting the great interest and rapid development of PP techniques in general. In contrast to using metagenomic sequences, metataxonomics has been the main focus because large data bases for different marker genes are readily available, and curated reference phylogenies exist. Restricting the analysis to single marker genes also scales well with ML programs, while using metagenomic read sequences with whole-genome references is often too computationally demanding. The assessment of the associated uncertainty of inferred placement positions is an important aspect during read identification. Several algorithms produce a measure of uncertainty; for ML-based methods, this measure is commonly derived from normalized likelihood values across all potential placement positions. We also presented such a measure for our alignment-free approach App-SpaM in Sec. 4.4 and, likewise, parametric and non-parametric bootstrapping have been proposed to assess uncertainty in APPLES [381]. Indicating placement uncertainty for short query reads is beneficial for guiding decisions in subsequent processing steps. For example, it may serve as a quality control by removing reads with high associated uncertainty. In addition to the uncertainty of a single query, it may be useful to describe *placement variety* among all queries. In a metataxonomic or metagenomic setting, placement variety also describes the within-sample diversity. The idea to utilize the query placement distribution for this purpose was already implemented and provided with pplacer by computing the *expected distance between placement locations* (EDPL) measure [12]. Subsequently, several other measures have been proposed to describe the diversity within or between samples based on phylogenetic placements [337]. In contrast to metagenomics, EDPL and associated measures can also describe the uncertainty of a species position when query sequences are whole genes or contigs from the same species. When a high degree of precision is mandatory, one possibility is to merge placement results from multiple PP programs. Joint placement is generally more accurate than single placements on their own [477].

It is commonly known that taxon sampling has a large impact on phylogenetic inference in general [478]; likewise, the composition of reference sequences has large implications on resulting placements. Accordingly, generating appropriate reference trees for phylogenetic placement has been an ongoing subject of study [333]. In general, a dense taxon sampling among the species that are expected to occur in the query sample is ideal. Placement algorithms simply assume that the taxon sampling in the provided reference tree is appropriately chosen by the end user for the placement task ahead. However, this assumption may be naive considering the increasing interest in and use of placement algorithms. Instead, programs may warn end users if the taxon sampling in the reference tree is insufficient, especially if a large number of query sequences is affected. Again, the uncertainty of a placement may be utilized for this purpose, as it also provides implicit information about the taxon sampling in the reference tree. Repeated placements at deep inner nodes with large uncertainty are a typical indicator of a deficit in taxon sampling. The necessity for a dense taxon sampling poses another risk for the application of placement programs: The use of large backbone trees is computationally challenging, especially if the reference sequences comprise parts of or complete genomes. Meanwhile, the continued increase in the amount of next- and third-generation sequencing data available [479] facilitates the use of large reference trees.

The memory demand and speed of most programs scale linearly with the number of reference sequences (assuming an existing reference alignment). Although the recent development of alignment-free programs has alleviated this problem to some degree, ML programs have been excluded from this progress until lately. In order to shift from a few hundred to tens of thousands of references, a sublinear scaling with respect to the number of references is required. Thus, simultaneous to the advancement of alignment-free PP programs, several *boosting* techniques have been proposed that attempt to scale ML-based placement programs to large reference phylogenies. Three prominent examples of such procedures are SCAMPP [480], pplacerDC [330], and pplacer-XR [481]. The general approach of all these approaches is to sample a subtree $\mathcal{T}'_{\text{ref}}$ from $\mathcal{T}_{\text{ref}}$ that has a high probability of containing the correct placement edge from $\mathcal{T}_{\text{ref}}$. After placing a query on $\mathcal{T}'_{\text{ref}}$, the results are transferred back to the original tree $\mathcal{T}_{\text{ref}}$. SCAMPP has been applied to trees with more than 200 000 reference sequences, and its results outperform those of APPLES and APPLES-II [480]. However, the disadvantage of these approaches is their continued reliance on multiple sequence alignments; accordingly, the evaluated sequence data span at most a single marker gene and have a mean alignment length of 2 570 bp. Based on the time and memory demands of App-SpaM for a large number of queries shown in Sec. 3.1.2, we also expect its seamless operation on large reference trees comprising single marker genes. However, experiments should be carried out that prove this hypothesis and quantify its actual speed. Furthermore, such efforts should include the placement on large reference trees that comprise whole genomes.

Only recently, the possibility of placing longer sequences has been tested more thoroughly, mostly driven by the development of novel PP programs. This development fits well into the growing number of metagenome assembled genomes and long-read sequencing techniques, and a whole new world of possibilities opens up that is being actively explored. Besides the update of large phylogenetic trees, other potential applications may be the mapping of genome scaffolds prior to assembly, the rooting of phylogenetic trees by placing species of an outgroup, or the estimation of contamination and completeness of assembled genomes as, for example, performed by BUSCO [482] and CheckM [338]. Although there has been a broad expansion of PP approaches and their applications, there is still much to discover: So far, all alignment-free methods are word-based methods because they utilize contiguous or non-contiguous k-mers of a fixed length. This causes inherent trade-offs, for example, between speed and accuracy for RAPPAS and APPLES. In this regard, we also determined multiple limitations of our approach App-SpaM, for example, regarding faulty distance estimates as discussed in Sec. 4.1. Furthermore, for a weight of $w = 12$, App-SpaM also encounters computational difficulties for large genome databases due to the quadratic increase in background matches, which can be alleviated by applying sampling or increasing the weight. However, as presented in Sec. 2.4, a broad range of other alignment-free approaches is still waiting to be utilized for phylogenetic placement and may promote new insights and new purposes. Only time will tell how PP algorithms and their wide-ranging possibilities will shape the field of phylogenomics in the years to come.

7.3 Conclusion

First envisaged for metataxonomic read identification, phylogenetic placement has found wide-ranging applications by now. This process has been driven by the development of a variety of alignment-based and alignment-free methods and associated tools. However, PP programs are most often applied by the same members of the scientific community who developed the programs in the first place. The uptake of PP programs by the wide range of potential end users and the number of metagenomic studies which utilize placement remains limited so far. One reason for this has been the effort, care, and knowledge of bioinformatic pipelines that has to be brought along to be able to practically apply PP tools. We believe that App-SpaM offers an easy entry point to the application of phylogenetic placement on diverse data sets because little preprocessing has to be performed. Another hurdle has been the insufficient presentation and explanation of phylogenetic placement, including its advantages and limitations, which recently have been covered comprehensively in a review [334]. Furthermore, the PP community should push to extend common standards for input and output file formats, openly accessible evaluation procedures, and standardized evaluation metrics. While the application of metataxonomic and metagenomic read identification has been adequately tested and presented in multiple instances, the usage of phylogenetic placement for other use cases requires more thorough testing to recognize under which circumstances placement produces reliable results. Such use cases include, for example, the update of large phylogenetic species trees under the presence of gene discordance, the detection of gene or species outliers, the estimation of contamination and completeness of assembled genomes, the binning of contigs, or the rooting of existing trees. Still, we believe that PP algorithms, together with existing frameworks that focus on phylogenetic data, already constitute an appealing group of well-tested methods to tackle a variety of metataxonomic, metagenomic, and phylogenomic research questions. Due to its unique approach, phylogenetic placement has already proven itself to be a valuable asset in the bioinformaticians toolbox, especially with regard to pressing issues in modern society: Phylogenetic placement is already regularly used in clinical studies [384, 483] and has been applied to the real-time tracking of pandemics [484]. Moreover, it may contribute to exploring and characterizing the diverse biospheres of Earth in a period where the rapid decline of animal and plant populations has taken place and will continue to take place in future decades and centuries [485, 486]. After all, there waits a lot to be discovered as life on Earth is complex and diverse.

Glossary

`A` Nucleobase adenine. 5, 6, 8, 9, 12, 17, 21, 59, 133

`C` Nucleobase cytosine. 5, 6, 8, 9, 12, 17, 21, 59, 133

`G` Nucleobase guanine. 5, 6, 8, 9, 12, 17, 21, 59, 133

`T` Nucleobase thymine. 5, 6, 8, 9, 12, 17, 21, 59, 133

`U` Nucleobase uracile. 6, 9

Σ Underlying alphabet of biological sequences. 9–12, 14, 17, 19, 21, 59, 133, 138

o Size of alphabet Σ. 9–11

S A biological sequence. 9, 20, 21, 98, 99, 133

S_q A query sequence. 14, 42, 44–47, 49–51, 55–58, 64, 78–80, 86, 90–95, 103–108, 111, 112, 114, 115, 119, 120, 122, 214

m Number of biological sequences under study. 9, 10, 18, 23, 25, 26, 28, 29, 32, 42–44, 50, 53, 55, 57, 58, 79, 92–94, 105, 111–114, 121, 209, 219

n Length of biological sequences under study. 9, 10, 13, 18–20, 32, 45, 98, 133–135, 153

k Length of continuous words. 9, 14, 18–20, 41, 133

P A binary pattern consisting of zeros and ones. 21–23, 55, 68, 98

w Number of match positions in spaced patterns. 54, 90, 92, 108

l Length of spaced patterns. 54, 98

t Filtering threshold for spaced-word matches. 14, 22, 23, 55–58, 89–92, 211

$\mathcal{T}_{\mathbf{ref}}$ Reference tree for phylogenetic placement. 42–50, 52–61, 73, 75, 81–83, 90, 92, 93, 97, 101, 102, 104–106, 108, 111, 143, 146, 203–205, 219

$\mathcal{T}_q$ Tree with an additionally placed sequence S_q. 46, 47, 93

$\mathcal{T}^*$ True underlying tree of a set of sequences. 48–50, 113, 117

$\mathcal{A}$ Multiple sequence alignment. 10

$\mathcal{A}_{\mathbf{ref}}$ Reference alignment for phylogenetic placement. 42, 45, 47, 48, 50, 60, 63

$\mathcal{A}^*$ True reference alignment. 50

$\mathbf{M}$ Distance matrix. 23, 28

M Substitution matrix for MSAs or filtering spaced-word matches. 11, 14, 22, 23, 89, 209

π Frequency vector of nucleotides. 11, 12

List of Acronyms

APPLES Accurate phylogenetic placement with least squares (software). 43, 46, 47, 50, 58, 60, 62, 64, 71, 72, 74, 75, 77, 78, 80, 83, 104, 120, 141, 145, 146, 183, 184

APPLES-II Accurate phylogenetic placement with least squares, version 2 (software). 46, 47, 146

DNA Deoxyribonucleic acid. 1, 3, 5–8, 10–15, 17, 18, 20, 22, 27, 28, 34–37, 41, 54, 59, 63, 85, 86, 132–134, 138, 140, 152

eND Expected node distance. 60, 64, 80, 81, 106, 142–144, 189–200

EPA Evolutionary placement algorithm (software). 42–46, 49, 50, 60–62, 71, 72, 74, 75, 80, 81, 83, 184

EPA-ng Fast, parallel, highly accurate maximum likelihood phylogenetic placement (software). 43–45, 49, 60–62, 71, 72, 74, 75, 80, 81, 95, 115–120, 123, 143, 153, 154, 184, 215–218

ML Maximum likelihood. 32, 33, 42–45, 47, 48, 54, 59, 60, 63, 64, 72, 77, 78, 80, 104, 106, 108, 110, 121, 126, 128, 140, 141, 143, 145

MP Maximum parsimony. 128, 140

MSA Multiple sequence alignment. 10, 11, 13–16, 31, 32, 45, 46, 48, 50, 58, 60, 74

ND Node distance. 51, 60, 63–73, 77, 80, 81, 83, 86, 97, 101, 106, 142–144, 185, 189, 190

PAC Pruning-based accuracy evaluation. 52, 60–63, 81, 155, 184, 203

PEWO Placement evaluation workflow. 51, 52, 59–63, 72, 75, 77, 80, 81, 87, 94, 100, 106, 120, 141–144, 203

PP Phylogenetic placement. 42–44, 46–50, 52, 53, 60, 63, 70–72, 74–77, 80, 81, 83, 108, 109, 141–145, 147, 152, 153, 183, 189–200, 203

pplacer Linear time maximum-likelihood and Bayesian phylogenetic placement (software). 42–46, 49, 50, 60, 62, 71, 72, 74, 75, 81, 145, 184

RAPPAS Rapid alignment-free phylogenetic placement via ancestral sequences (software). 43, 45, 46, 60, 62, 70–72, 74, 75, 77, 80, 81, 83, 104, 105, 108, 143, 146, 183, 184

RES Resources evaluation. 52, 60–63, 75, 79, 155, 184

RNA Ribonucleic acid. 6–10, 14, 15, 22, 35, 37, 41, 63, 140

List of Figures

List of Tables

Bibliography

1. Dodd, M. S. *et al.* Evidence for early life in Earth's oldest hydrothermal vent precipitates. *Nature* **543,** 60–64 (2017).
2. Betts, H. C. *et al.* Integrated genomic and fossil evidence illuminates life's early evolution and eukaryote origin. *Nature Ecology & Evolution* **2,** 1556–1562 (2018).
3. Mora, C., Tittensor, D. P., Adl, S., Simpson, A. G. B. & Worm, B. How Many Species Are There on Earth and in the Ocean? *PLOS Biology* **9,** e1001127 (2011).
4. Locey, K. J. & Lennon, J. T. Scaling laws predict global microbial diversity. *Proceedings of the National Academy of Sciences* **113,** 5970–5975 (2016).
5. Ochman, H., Lawrence, J. G. & Groisman, E. A. Lateral gene transfer and the nature of bacterial innovation. *Nature* **405,** 299–304 (2000).
6. Wooley, J. C., Godzik, A. & Friedberg, I. A Primer on Metagenomics. *PLOS Computational Biology* **6,** e1000667 (2010).
7. Breitwieser, F. P., Lu, J. & Salzberg, S. L. A review of methods and databases for metagenomic classification and assembly. *Briefings in Bioinformatics* **20,** 1125–1136 (2019).
8. Fuentes-Pardo, A. P. & Ruzzante, D. E. Whole-genome sequencing approaches for conservation biology: Advantages, limitations and practical recommendations. *Molecular Ecology* **26,** 5369–5406 (2017).
9. Yang, C. *et al.* A review of computational tools for generating metagenome-assembled genomes from metagenomic sequencing data. *Computational and Structural Biotechnology Journal* **19,** 6301–6314 (2021).
10. Berger, S. A., Krompass, D. & Stamatakis, A. Performance, Accuracy, and Web Server for Evolutionary Placement of Short Sequence Reads under Maximum Likelihood. *Systematic Biology* **60,** 291–302 (2011).
11. Leimeister, C.-A., Sohrabi-Jahromi, S. & Morgenstern, B. Fast and accurate phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics* **33,** 971–979 (2017).
12. Matsen, F. A., Kodner, R. B. & Armbrust, E. V. pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. *BMC Bioinformatics* **11,** 538 (2010).
13. Balaban, M., Sarmashghi, S. & Mirarab, S. APPLES: Fast Distance-based Phylogenetic Placement. *Systematic Biology* **69,** 566–578 (2020).
14. Linard, B., Swenson, K. & Pardi, F. Rapid alignment-free phylogenetic identification of metagenomic sequences. *Bioinformatics* **35,** 3303–3312 (2019).
15. Barbera, P. *et al.* EPA-ng: Massively Parallel Evolutionary Placement of Genetic Sequences. *Systematic Biology* **68,** 365–369 (2019).
16. Woese, C. R., Kandler, O & Wheelis, M. L. Towards a natural system of organisms: proposal for the domains Archaea, Bacteria, and Eucarya. *Proceedings of the National Academy of Sciences* **87,** 4576–4579 (1990).
17. Whitman, W. B., Coleman, D. C. & Wiebe, W. J. Prokaryotes: the unseen majority. *Proceedings of the National Academy of Sciences* **95,** 6578–6583 (1998).

18. Ruggiero, M. A. *et al.* A Higher Level Classification of All Living Organisms. *PLOS ONE* **10,** e0119248 (2015).
19. Dahm, R. Discovering DNA: Friedrich Miescher and the early years of nucleic acid research. *Human Genetics* **122,** 565–581 (2008).
20. Dahm, R. Friedrich Miescher and the discovery of DNA. *Developmental Biology* **278,** 274–288 (2005).
21. Watson, J. D. & Crick, F. H. C. The Structure of Dna. *Cold Spring Harbor Symposia on Quantitative Biology* **18,** 123–131 (1953).
22. Selzer, P. M., Marhöfer, R. J. & Koch, O. *Applied bioinformatics: An introduction* ISBN: 978-3-319-68301-0 (Springer, 2018).
23. Statello, L., Guo, C.-J., Chen, L.-L. & Huarte, M. Gene regulation by long non-coding RNAs and its biological functions. *Nature Reviews Molecular Cell Biology* **22,** 96–118 (2021).
24. Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C. & Zody, M. C. Initial sequencing and analysis of the human genome. *Nature* **409,** 860–921 (2001).
25. Ezkurdia, I. *et al.* Multiple evidence strands suggest that there may be as few as 19 000 human protein-coding genes. *Human Molecular Genetics* **23,** 5866–5878 (2014).
26. Salzberg, S. L. Open questions: How many genes do we have? *BMC Biology* **16,** 94 (2018).
27. Rogozin, I. B. *et al.* Congruent evolution of different classes of non-coding DNA in prokaryotic genomes. *Nucleic Acids Research* **30,** 4264–4271 (2002).
28. Thomas, C. M. & Nielsen, K. M. Mechanisms of, and Barriers to, Horizontal Gene Transfer between Bacteria. *Nature Reviews Microbiology* **3,** 711–721 (2005).
29. Andersson, J. O. Lateral gene transfer in eukaryotes. *Cellular and Molecular Life Sciences CMLS* **62,** 1182–1197 (2005).
30. Keeling, P. J. & Palmer, J. D. Horizontal gene transfer in eukaryotic evolution. *Nature Reviews Genetics* **9,** 605–618 (2008).
31. Husnik, F. & McCutcheon, J. P. Functional horizontal gene transfer from bacteria to eukaryotes. *Nature Reviews Microbiology* **16,** 67–79 (2018).
32. Brooks, P. J., Cheng, T.-F. & Cooper, L. Do all of the neurologic diseases in patients with DNA repair gene mutations result from the accumulation of DNA damage? *DNA Repair* **7,** 834–848 (2008).
33. Branum, M. E., Reardon, J. T. & Sancar, A. DNA repair excision nuclease attacks undamaged DNA: A potential source of spontaneous mutations. *Journal of Biological Chemistry* **276,** 25421–25426 (2001).
34. Vol'kenshtein, M. V. Probabilities of transversions and transitions. *Molekuliarnaia Biologiia* **10,** 605–608 (1976).
35. Griffiths, A. J. F., Miller, J. H., Suzuki, D. T., Lewontin, R. C. & Gelbart, W. M. *Introduction to Genetic Analysis* ISBN: 978-0716799023 (W. H. Freeman, 2000).
36. Linder, C. R. & Rieseberg, L. H. Reconstructing patterns of reticulate evolution in plants. *American Journal of Botany* **91,** 1700–1708 (2004).
37. *Sequence Alignment: Methods, Models, Concepts, and Strategies* 0 Edition (ed Rosenberg, M. S.) ISBN: 978-0-520-25697-2 (University of California Press, 2009).
38. Margoliash, E. Primary structure and evolution of cytochrome c. *Proceedings of the National Academy of Sciences* **50,** 672–679 (1963).
39. Suárez-Díaz, E. The Long and Winding Road of Molecular Data in Phylogenetic Analysis. *Journal of the History of Biology* **47,** 443–478 (2014).
40. Garcia-Sancho, M. *Biology, computing, and the history of molecular sequencing: from proteins to DNA, 1945-2000* (Springer, 2012).

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

42. Chojnacki, S., Cowley, A., Lee, J., Foix, A. & Lopez, R. Programmatic access to bioinformatics tools from EMBL-EBI update: 2017. *Nucleic Acids Research* **45,** W550–W553 (2017).

43. Henikoff, S & Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences* **89,** 10915–10919 (1992).

44. Tavaré, S. Some probabilistic and statistical problems in the analysis of DNA sequences. *Lectures on mathematics in the life sciences* **17,** 57–86 (1986).

45. Jukes, T. H. & Cantor, C. R. *Evolution of Protein Molecules* in *Mammalian Protein Metabolism* (ed Munro, H. N.) (Academic Press, 1969), 21–132. ISBN: 978-1-4832-3211-9.

46. Kimura, M. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution* **16,** 111–120 (1980).

47. Zharkikh, A. Estimation of evolutionary distances between nucleotide sequences. *Journal of Molecular Evolution* **39,** 315–329 (1994).

48. Tamura, K & Nei, M. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Molecular Biology and Evolution* **10,** 512–526 (1993).

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

50. Hasegawa, M., Kishino, H. & Yano, T.-a. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution* **22,** 160–174 (1985).

51. Felsenstein, J. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution* **17,** 368–376 (1981).

52. Chan, K.-F., Koukouravas, S., Yeo, J. Y., Koh, D. W.-S. & Gan, S. K.-E. Probability of change in life: Amino acid changes in single nucleotide substitutions. *Biosystems* **193-194,** 104135 (2020).

53. Dayhoff, M. O., Schwartz, R. M. & Orcutt, B. *A Model of Evolutionary Change in Proteins* in *Atlas of protein sequence and structure* **5** (National Biomedical Research Foundation, 1978), 89–99.

54. Müller, T. & Vingron, M. Modeling Amino Acid Replacement. *Journal of Computational Biology* **7,** 761–776 (2000).

55. Le, S. Q. & Gascuel, O. An Improved General Amino Acid Replacement Matrix. *Molecular Biology and Evolution* **25,** 1307–1320 (2008).

56. Whelan, S. & Goldman, N. A General Empirical Model of Protein Evolution Derived from Multiple Protein Families Using a Maximum-Likelihood Approach. *Molecular Biology and Evolution* **18,** 691–699 (2001).

57. Bevan, R. B., Bryant, D. & Lang, B. F. Accounting for Gene Rate Heterogeneity in Phylogenetic Inference. *Systematic Biology* **56,** 194–205 (2007).

58. Lange, K. *Mathematical and statistical methods for genetic analysis* (Springer, 2002).

59. Wang, L. & Jiang, T. On the Complexity of Multiple Sequence Alignment. *Journal of Computational Biology* **1,** 337–348 (1994).

60. Masek, W. J. & Paterson, M. S. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences* **20,** 18–31 (1980).

61. Smith, T. F. & Waterman, M. S. Identification of common molecular subsequences. *Journal of Molecular Biology* **147,** 195–197 (1981).

62. Liu, Y., Huang, W., Johnson, J. & Vaidya, S. *GPU accelerated Smith-Waterman* in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Springer, 2006). ISBN: 3540343857.

63. Liu, Y. & Schmidt, B. *SWAPHI: Smith-waterman protein database search on Xeon Phi coprocessors* in *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors* (IEEE, 2014). ISBN: 9781479936090.

64. Sandes, E. F. D. O. & De Melo, A. C. M. *Smith-Waterman alignment of huge sequences with GPU in linear space* in *Proceedings - 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011* (IEEE, 2011). ISBN: 9780769543857.

65. Farrar, M. Striped Smith–Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics* **23,** 156–161 (2007).

66. Page, A. J. *et al.* SNP-sites: rapid efficient extraction of SNPs from multi-FASTA alignments. *Microbial Genomics* **2,** e000056 (2016).

67. Chatzou, M. *et al.* Multiple sequence alignment modeling: methods and applications. *Briefings in Bioinformatics* **17,** 1009–1023 (2016).

68. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *Journal of Molecular Biology* **215,** 403–410 (1990).

69. Altschul, S. F. *et al.* Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* **25,** 3389–3402 (1997).

70. Kent, W. J. BLAT—The BLAST-Like Alignment Tool. *Genome Research* **12,** 656–664 (2002).

71. Van Nguyen, H. & Lavenier, D. PLAST: parallel local alignment search tool for database comparison. *BMC Bioinformatics* **10,** 329 (2009).

72. Morgulis, A. *et al.* Database indexing for production MegaBLAST searches. *Bioinformatics* **24,** 1757–1764 (2008).

73. Boratyn, G. M., Thierry-Mieg, J., Thierry-Mieg, D., Busby, B. & Madden, T. L. Magic-BLAST, an accurate RNA-seq aligner for long and short reads. *BMC Bioinformatics* **20,** 405 (2019).

74. Pearson, W. R. & Lipman, D. J. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences* **85,** 2444–2448 (1988).

75. Edgar, R. C. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics* **26,** 2460–2461 (2010).

76. Ma, B., Tromp, J. & Li, M. PatternHunter: faster and more sensitive homology search. *Bioinformatics* **18,** 440–445 (2002).

77. Feng, D.-F. & Doolittle, R. F. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of Molecular Evolution* **25,** 351–360 (1987).

78. Thompson, J. D., Higgins, D. G. & Gibson, T. J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research* **22,** 4673–4680 (1994).

79. Thompson, J. D., Gibson, T. J. & Higgins, D. G. Multiple Sequence Alignment Using ClustalW and ClustalX. *Current Protocols in Bioinformatics* **00,** 2.3.1–2.3.22 (2003).

80. Morgenstern, B., Frech, K., Dress, A. & Werner, T. DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics* **14,** 290–294 (1998).

81. Subramanian, A. R., Weyer-Menkhoff, J., Kaufmann, M. & Morgenstern, B. DIALIGN-T: An improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics* **6,** 66 (2005).

82. Katoh, K., Misawa, K., Kuma, K. & Miyata, T. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research* **30,** 3059–3066 (2002).

83. Hirosawa, M., Totoki, Y., Hoshida, M. & Ishikawa, M. Comprehensive study on iterative algorithms of multiple sequence alignment. *Bioinformatics* **11,** 13–18 (1995).

84. Liu, Y., Schmidt, B. & Maskell, D. L. MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities. *Bioinformatics* **26,** 1958–1964 (2010).

85. Katoh, K., Kuma, K.-i., Toh, H. & Miyata, T. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research* **33,** 511–518 (2005).

86. Katoh, K. & Toh, H. Parallelization of the MAFFT multiple sequence alignment program. *Bioinformatics* **26,** 1899–1900 (2010).

87. Katoh, K. & Standley, D. M. MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability. *Molecular Biology and Evolution* **30,** 772–780 (2013).

88. Rozewicki, J., Li, S., Amada, K. M., Standley, D. M. & Katoh, K. MAFFT-DASH: integrated protein sequence and structural alignment. *Nucleic Acids Research* **47,** W5–W10 (2019).

89. Notredame, C., Higgins, D. G. & Heringa, J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology* **302,** 205–217 (2000).

90. Edgar, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research* **32,** 1792–1797 (2004).

91. Eddy, S. R. *Multiple alignment using hidden Markov models.* in *Ismb* **3** (AAAI, 1995), 114–120.

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

93. Krogh, A., Brown, M., Mian, I. S., Sjölander, K. & Haussler, D. Hidden Markov Models in Computational Biology: Applications to Protein Modeling. *Journal of Molecular Biology* **235,** 1501–1531 (1994).

94. Eddy, S. R. Profile hidden Markov models. *Bioinformatics* **14,** 755–763 (1998).

95. Finn, R., Clements, J. & Eddy, S. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Research* **39,** W29–W37 (2011).

96. Sievers, F. *et al.* Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology* **7,** 539 (2011).

97. Steinegger, M. *et al.* HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics* **20,** 473 (2019).

98. Söding, J. Protein homology detection by HMM–HMM comparison. *Bioinformatics* **21,** 951–960 (2005).

99. Löytynoja, A. & Goldman, N. An algorithm for progressive multiple alignment of sequences with insertions. *Proceedings of the National Academy of Sciences* **102,** 10557–10562 (2005).

100. Löytynoja, A., Vilella, A. J. & Goldman, N. Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm. *Bioinformatics* **28,** 1684–1691 (2012).

101. Liu, K., Raghavan, S., Nelesen, S., Linder, C. R. & Warnow, T. Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees. *Science* **324,** 1561–1564 (2009).

102. Notredame, C. Recent Evolutions of Multiple Sequence Alignment Algorithms. *PLOS Computational Biology* **3,** e123 (2007).

103. Chowdhury, B. & Garai, G. A review on multiple sequence alignment from the perspective of genetic algorithm. *Genomics* **109,** 419–431 (2017).

104. Tamura, K. Estimation of the number of nucleotide substitutions when there are strong transition-transversion and G+C-content biases. *Molecular Biology and Evolution* **9,** 678–687 (1992).

105. Comeron, J. M. K-Estimator: calculation of the number of nucleotide substitutions per site and the confidence intervals. *Bioinformatics* **15,** 763–764 (1999).

106. Nei, M. & Tajima, F. Maximum likelihood estimation of the number of nucleotide substitutions from restriction sites data. *Genetics* **105,** 207–217 (1983).

107. Posada, D. & Crandall, K. A. Selecting the Best-Fit Model of Nucleotide Substitution. *Systematic Biology* **50,** 580–601 (2001).

108. Kishino, H., Thorne, J. L. & Bruno, W. J. Performance of a Divergence Time Estimation Method under a Probabilistic Model of Rate Evolution. *Molecular Biology and Evolution* **18,** 352–361 (2001).

109. Thorne, J. L. & Kishino, H. Divergence Time and Evolutionary Rate Estimation with Multilocus Data. *Systematic Biology* **51,** 689–702 (2002).

110. Bordoli, L. *et al.* Protein structure homology modeling using SWISS-MODEL workspace. *Nature Protocols* **4,** 1–13 (2009).

111. Martí-Renom, M. A. *et al.* Comparative Protein Structure Modeling of Genes and Genomes. *Annual Review of Biophysics and Biomolecular Structure* **29,** 291–325 (2000).

112. Shihab, H. A. *et al.* Predicting the Functional, Molecular, and Phenotypic Consequences of Amino Acid Substitutions using Hidden Markov Models. *Human Mutation* **34,** 57–65 (2013).

113. Pierri, C. L., Parisi, G. & Porcelli, V. Computational approaches for protein function prediction: A combined strategy from multiple sequence alignment to molecular docking-based virtual screening. *Biochimica et Biophysica Acta (BBA) - Proteins and Proteomics* **1804,** 1695–1712 (2010).

114. Hannenhalli, S. S. & Russell, R. B. Analysis and prediction of functional sub-types from protein sequence alignments11Edited by J. Thornton. *Journal of Molecular Biology* **303,** 61–76 (2000).

115. Hashim, F. A., Mabrouk, M. S. & Al-Atabany, W. Review of Different Sequence Motif Finding Algorithms. *Avicenna Journal of Medical Biotechnology* **11,** 130–148 (2019).

116. Das, M. K. & Dai, H.-K. A survey of DNA motif finding algorithms. *BMC Bioinformatics* **8,** S21 (2007).

117. Gross, S. S. & Brent, M. R. Using Multiple Alignments to Improve Gene Prediction. *Journal of Computational Biology* **13,** 379–393 (2006).

118. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology* **10,** R25 (2009).

119. Goenka, S. D., Turakhia, Y., Paten, B. & Horowitz, M. *SegAlign: A Scalable GPU-Based Whole Genome Aligner* in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE, 2020), 1–13.

120. Bushnell, B. BBMap: A Fast, Accurate, Splice-Aware Aligner (2014).

121. Stamatakis, A., Ludwig, T. & Meier, H. RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics* **21,** 456–463 (2005).

122. Hall, B. G. Building Phylogenetic Trees from Molecular Data with MEGA. *Molecular Biology and Evolution* **30,** 1229–1235 (2013).

123. Huelsenbeck, J. P. & Ronquist, F. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* **17,** 754–755 (2001).

124. Liu, K. *et al.* SATé-II: Very Fast and Accurate Simultaneous Estimation of Multiple Sequence Alignments and Phylogenetic Trees. *Systematic Biology* **61,** 90 (2012).

125. Kovaka, S. *et al.* Transcriptome assembly from long-read RNA-seq alignments with StringTie2. *Genome Biology* **20,** 278 (2019).

126. Ayling, M., Clark, M. D. & Leggett, R. M. New approaches for metagenome assembly with short reads. *Briefings in Bioinformatics* **21,** 584–594 (2020).

127. Kim, D., Langmead, B. & Salzberg, S. L. HISAT: a fast spliced aligner with low memory requirements. *Nature Methods* **12,** 357–360 (2015).

128. Dobin, A. *et al.* STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29,** 15–21 (2013).

129. Schulz, T., Wittler, R., Rahmann, S., Hach, F. & Stoye, J. Detecting high-scoring local alignments in pangenome graphs. *Bioinformatics* **37,** 2266–2274 (2021).

130. Chao, K. M. & Miller, W. Linear-space algorithms that build local alignments from fragments. *Algorithmica* **13,** 106–134 (1995).

131. Kehr, B., Weese, D. & Reinert, K. STELLAR: fast and exact local alignments. *BMC Bioinformatics* **12,** S15 (2011).

132. Paten, B. *et al.* Cactus: Algorithms for genome multiple sequence alignment. *Genome Research* **21,** 1512–1528 (2011).

133. Poliakov, A., Foong, J., Brudno, M. & Dubchak, I. GenomeVISTA—an integrated software package for whole-genome alignment and visualization. *Bioinformatics* **30,** 2654–2655 (2014).

134. Couronne, O. *et al.* Strategies and Tools for Whole-Genome Alignments. *Genome Research* **13,** 73–80 (2003).

135. Borozan, I., Watt, S. & Ferretti, V. Integrating alignment-based and alignment-free sequence similarity measures for biological sequence classification. *Bioinformatics* **31,** 1396–1404 (2015).

136. Earl, D. *et al.* Alignathon: a competitive assessment of whole-genome alignment methods. *Genome Research* **24,** 2077–2089 (2014).

137. Zielezinski, A., Vinga, S., Almeida, J. & Karlowski, W. M. Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biology* **18,** 186 (2017).

138. Vinga, S. Information theory applications for biological sequence analysis. *Briefings in Bioinformatics* **15,** 376–389 (2014).

139. Höhl, M. & Ragan, M. A. Is Multiple-Sequence Alignment Required for Accurate Inference of Phylogeny? *Systematic Biology* **56,** 206–221 (2007).

140. Giancarlo, R., Rombo, S. E. & Utro, F. Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies. *Briefings in Bioinformatics* **15,** 390–406 (2014).

141. Chen, X., Kwong, S. & Li, M. A Compression Algorithm for DNA Sequences and Its Applications in Genome Comparison. *Genome Informatics* **10,** 51–61 (1999).

142. Mantaci, S., Restivo, A., Rosone, G. & Sciortino, M. *An Extension of the Burrows Wheeler Transform and Applications to Sequence Comparison and Data Compression* in *Combinatorial Pattern Matching* (eds Apostolico, A., Crochemore, M. & Park, K.) (Springer, 2005), 178–189. ISBN: 978-3-540-31562-9.

143. Hosseini, M., Pratas, D., Morgenstern, B. & Pinho, A. J. Smash++: an alignment-free and memory-efficient tool to find genomic rearrangements. *GigaScience* **9,** giaa048 (2020).

144. Kullback, S. & Leibler, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics* **22,** 79–86 (1951).

145. Ulitsky, I., Burstein, D., Tuller, T. & Chor, B. The Average Common Substring Approach to Phylogenomic Reconstruction. *Journal of Computational Biology* **13,** 336–350 (2006).

146. Weiner, P. *Linear pattern matching algorithms* in *14th Annual Symposium on Switching and Automata Theory* (IEEE, 1973).

147. Vinga, S. Biological sequence analysis by vector-valued functions: revisiting alignment-free methodologies for DNA and protein classification. *Advanced Computational Methods for Biocomputing and Bioimaging.* (2007).

148. Wood, D. & Salzberg, S. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology* **15,** R46 (2014).

149. Ondov, B. D. *et al.* Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology* **17,** 132 (2016).

150. Marçais, G., DeBlasio, D., Pandey, P. & Kingsford, C. Locality-sensitive hashing for the edit distance. *Bioinformatics* **35,** i127–i135 (2019).

151. Jain, M. *et al.* Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology* **36,** 338–345 (2018).

152. Berlin, K. *et al.* Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature Biotechnology* **33,** 623–630 (2015).

153. Mounsef, N., Karam, L., Lacroix, Z. & Legendre, C. *A low-complexity probabilistic genome assembly based on hashing functions with SNP detection* in *2008 IEEE International Workshop on Genomic Signal Processing and Statistics* (IEEE, 2008), 1–4. ISBN: 978-1-4244-2371-2.

154. Haubold, B., Pierstorff, N., Möller, F. & Wiehe, T. Genome comparison without alignment using shortest unique substrings. *BMC Bioinformatics* **6,** 123 (2005).

155. Leimeister, C.-A. & Morgenstern, B. kmacs: the k -mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics* **30,** 2000–2008 (2014).

156. Pinho, A. J., Ferreira, P. J., Garcia, S. P. & Rodrigues, J. M. On finding minimal absent words. *BMC Bioinformatics* **10,** 137 (2009).

157. Yang, L., Zhang, X., Wang, T. & Zhu, H. Large Local Analysis of the Unaligned Genome and Its Application. *Journal of Computational Biology* **20,** 19–29 (2013).

158. Didier, G., Corel, E., Laprevotte, I., Grossmann, A. & Landès-Devauchelle, C. Variable length local decoding and alignment-free sequence comparison. *Theoretical Computer Science* **462,** 1–11 (2012).

159. Comin, M. & Verzotto, D. Alignment-free phylogeny of whole genomes using underlying subwords. *Algorithms for Molecular Biology* **7,** 34 (2012).

160. Vinga, S. & Almeida, J. Alignment-free sequence comparison—a review. *Bioinformatics* **19,** 513–523 (2003).

161. Haubold, B. Alignment-free phylogenetics and population genetics. *Briefings in Bioinformatics* **15,** 407–418 (2014).

162. Yi, H. & Jin, L. Co-phylog: an assembly-free phylogenomic approach for closely related organisms. *Nucleic Acids Research* **41,** e75 (2013).

163. Haubold, B., Klötzl, F. & Pfaffelhuber, P. andi: Fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics* **31,** 1169–1175 (2015).

164. Leimeister, C.-A., Sohrabi-Jahromi, S. & Morgenstern, B. Fast and Accurate Phylogeny Reconstruction using Filtered Spaced-Word Matches. *Bioinformatics* **33,** 971–979 (2017).

165. Hahn, L., Leimeister, C.-A., Ounit, R., Lonardi, S. & Morgenstern, B. *rasbhari*: optimizing spaced seeds for database searching, read mapping and alignment-free sequence comparison. *PLOS Computational Biology* **12,** e1005107 (2016).

166. Lau, A.-K., Dörrer, S., Leimeister, C.-A., Bleidorn, C. & Morgenstern, B. Read-SpaM: assembly-free and alignment-free comparison of bacterial genomes with low sequencing coverage. *BMC Bioinformatics* **20,** 638 (2019).

167. Leimeister, C.-A. *et al.* Prot-SpaM: fast alignment-free phylogeny reconstruction based on whole-proteome sequences. *GigaScience* **8,** giy148 (2019).

168. Dencker, T. *et al.* 'Multi-SpaM': a maximum-likelihood approach to phylogeny reconstruction using multiple spaced-word matches and quartet trees. *NAR Genomics and Bioinformatics* **2,** lqz013 (2020).

169. Snir, S. & Rao, S. Quartet MaxCut: A fast algorithm for amalgamating quartet trees. *Molecular Phylogenetics and Evolution* **62,** 1–8 (2012).

170. Röhling, S. *et al.* The number of k-mer matches between two DNA sequences as a function of k and applications to estimate phylogenetic distances. *PLOS ONE* **15,** e0228070 (2020).

171. Leimeister, C.-A., Dencker, T. & Morgenstern, B. Accurate multiple alignment of distantly related genome sequences using Filtered Spaced Word Matches as anchor points. *Bioinformatics* **35,** 211–218 (2019).

172. Nazeen, S. & Berger, B. *Carnelian: alignment-free functional binning and abundance estimation of metagenomic reads* 2018.

173. Kouchaki, S., Tapinos, A. & Robertson, D. L. A signal processing method for alignment-free metagenomic binning: multi-resolution genomic binary patterns. *Scientific Reports* **9,** 2159 (2019).

174. Pham, D.-T., Gao, S. & Phan, V. An accurate and fast alignment-free method for profiling microbial communities. *Journal of Bioinformatics and Computational Biology* **15,** 1740001 (2017).

175. Menzel, P., Ng, K. L. & Krogh, A. Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nature Communications* **7,** 11257 (2016).

176. Jaffe, D. B. *et al.* Whole-Genome Sequence Assembly for Mammalian Genomes: Arachne 2. *Genome Research* **13,** 91–96 (2003).

177. Blanke, M. & Morgenstern, B. App-SpaM: phylogenetic placement of short reads without sequence alignment. *Bioinformatics Advances* **1,** vbab027 (2021).

178. Laczny, C. C., Pinel, N., Vlassis, N. & Wilmes, P. Alignment-free Visualization of Metagenomic Data by Nonlinear Dimension Reduction. *Scientific Reports* **4,** 4516 (2014).

179. Sun, S., Xu, L., Zou, Q. & Wang, G. BP4RNAseq: a babysitter package for retrospective and newly generated RNA-seq data analyses using both alignment-based and alignment-free quantification method. *Bioinformatics* **37,** 1319–1321 (2021).

180. Heyne, S., Costa, F., Rose, D. & Backofen, R. GraphClust: alignment-free structural clustering of local RNA secondary structures. *Bioinformatics* **28,** i224–i232 (2012).

181. Lachmann, A., Clarke, D. J. B., Torre, D., Xie, Z. & Ma'ayan, A. Interoperable RNA-Seq analysis in the cloud. *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms. Transcriptional Profiles and Regulatory Gene Networks* **1863,** 194521 (2020).

182. Pratas, D., Silva, R. M., Pinho, A. J. & Ferreira, P. J. S. G. An alignment-free method to find and visualise rearrangements between pairs of DNA sequences. *Scientific Reports* **5,** 10203 (2015).

183. Haubold, B., Domazet-Lošo, M. & Wiehe, T. *An Alignment-Free Distance Measure for Closely Related Genomes* in *Comparative Genomics* (eds Nelson, C. E. & Vialette, S.) (Springer, 2008), 87–99. ISBN: 978-3-540-87989-3.

184. Pinello, L., Lo Bosco, G. & Yuan, G.-C. Applications of alignment-free methods in epigenomics. *Briefings in Bioinformatics* **15,** 419–430 (2014).

185. Kantorovitz, M. R., Robinson, G. E. & Sinha, S. A statistical method for alignment-free comparison of regulatory sequences. *Bioinformatics* **23,** i249–i255 (2007).

186. Seo, S., Oh, M., Park, Y. & Kim, S. DeepFam: deep learning based alignment-free method for protein family modeling and prediction. *Bioinformatics* **34,** i254–i262 (2018).

187. Hall, B. G. SNP-Associations and Phenotype Predictions from Hundreds of Microbial Genomes without Genome Alignments. *PLOS ONE* **9,** e90490 (2014).

188. Kapli, P., Yang, Z. & Telford, M. J. Phylogenetic tree building in the genomic age. *Nature Reviews Genetics* **21,** 428–444 (2020).

189. Daniell, H., Lin, C.-S., Yu, M. & Chang, W.-J. Chloroplast genomes: diversity, evolution, and applications in genetic engineering. *Genome Biology* **17,** 134 (2016).

190. Gao, L., Su, Y.-J. & Wang, T. Plastid genome sequencing, comparative genomics, and phylogenomics: Current status and prospects. *Journal of Systematics and Evolution* **48,** 77–93 (2010).

191. Rubinoff, D. & Holland, B. S. Between Two Extremes: Mitochondrial DNA is neither the Panacea nor the Nemesis of Phylogenetic and Taxonomic Inference. *Systematic Biology* **54,** 952–961 (2005).

192. Som, A. Causes, consequences and solutions of phylogenetic incongruence. *Briefings in Bioinformatics* **16,** 536–548 (2015).

193. Tringe, S. G. & Hugenholtz, P. A renaissance for the pioneering 16S rRNA gene. *Current Opinion in Microbiology. Antimicrobials/Genomics* **11,** 442–446 (2008).

194. Zafeiropoulos, H. *et al.* PEMA: a flexible Pipeline for Environmental DNA Metabarcoding Analysis of the 16S/18S ribosomal RNA, ITS, and COI marker genes. *GigaScience* **9,** giaa022 (2020).

195. Kryvenda, A., Rybalka, N., Wolf, M. & Friedl, T. Species distinctions among closely related strains of Eustigmatophyceae (Stramenopiles) emphasizing ITS2 sequence-structure data: Eustigmatos and Vischeria. *European Journal of Phycology* **53,** 471–491 (2018).

196. Huson, D. H. & Bryant, D. Application of Phylogenetic Networks in Evolutionary Studies. *Molecular Biology and Evolution* **23,** 254–267 (2006).

197. Huson, D. H. & Scornavacca, C. A Survey of Combinatorial Methods for Phylogenetic Networks. *Genome Biology and Evolution* **3,** 23–35 (2011).

198. Jin, G., Nakhleh, L., Snir, S. & Tuller, T. Maximum likelihood of phylogenetic networks. *Bioinformatics* **22,** 2604–2611 (2006).

199. Kannan, L. & Wheeler, W. C. Maximum Parsimony on Phylogenetic networks. *Algorithms for Molecular Biology* (2012).

200. Solís-Lemus, C., Bastide, P. & Ané, C. PhyloNetworks: A Package for Phylogenetic Networks. *Molecular Biology and Evolution* **34,** 3292–3298 (2017).

201. Wen, D., Yu, Y., Zhu, J. & Nakhleh, L. Inferring Phylogenetic Networks Using PhyloNet. *Systematic Biology* **67,** 735–740 (2018).

202. Schliep, K., Potts, A. A., Morrison, D. A. & Grimm, G. W. *Intertwining phylogenetic trees and networks* tech. rep. e2054v1 (PeerJ Inc., 2016).

203. Moret, B. M. E. & Warnow, T. *Advances in Phylogeny Reconstruction from Gene Order and Content Data* in *Methods in Enzymology* **395** (Academic Press, 2005), 673–700.

204. Sokal, R. R. A statistical method for evaluating systematic relationships. *Univ Kans Sci Bull* (1958).

205. Saitou, N. & Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* **4,** 406–425 (1987).

206. Desper, R. & Gascuel, O. *The minimum evolution distance-based approach of phylogenetic inference.* ISBN: 978-0199231348 (Clarendon Press, 2007).

207. Pauplin, Y. Direct Calculation of a Tree Length Using a Distance Matrix. *Journal of Molecular Evolution* **51,** 41–47 (2000).

208. Desper, R. & Gascuel, O. Theoretical Foundation of the Balanced Minimum Evolution Method of Phylogenetic Inference and Its Relationship to Weighted Least-Squares Tree Fitting. *Molecular Biology and Evolution* **21,** 587–598 (2004).

209. Lefort, V., Desper, R. & Gascuel, O. FastME 2.0: A Comprehensive, Accurate, and Fast Distance-Based Phylogeny Inference Program. *Molecular Biology and Evolution* **32,** 2798–2800 (2015).

210. Gascuel, O. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular Biology and Evolution* **14,** 685–695 (1997).

211. Criscuolo, A. & Gascuel, O. Fast NJ-like algorithms to deal with incomplete distance matrices. *BMC Bioinformatics* **9,** 166 (2008).

212. Liu, L. & Yu, L. Estimating Species Trees from Unrooted Gene Trees. *Systematic Biology* **60,** 661–667 (2011).

213. Vachaspati, P. & Warnow, T. ASTRID: Accurate Species TRees from Internode Distances. *BMC Genomics* **16,** S3 (2015).

214. Cavalli-Sforza, L. L. & Edwards, A. W. F. Phylogenetic analysis. Models and estimation procedures. *American Journal of Human Genetics* **19,** 233–257 (1967).

215. Rzhetsky, A. & Nei, M. Statistical properties of the ordinary least-squares, generalized least-squares, and minimum-evolution methods of phylogenetic inference. *Journal of Molecular Evolution* **35,** 367–375 (1992).

216. Felsenstein, J. An Alternating Least Squares Approach to Inferring Phylogenies from Pairwise Distances. *Systematic Biology* **46,** 101–111 (1997).

217. Symonds, M. R. E. & Blomberg, S. P. *A Primer on Phylogenetic Generalised Least Squares* in *Modern Phylogenetic Comparative Methods and Their Application in Evolutionary Biology: Concepts and Practice* (ed Garamszegi, L. Z.) (Springer, 2014), 105–130. ISBN: 978-3-662-43550-2.

218. Takezaki, N. & Nei, M. Inconsistency of the maximum parsimony method when the rate of nucleotide substitution is constant. *Journal of Molecular Evolution* **39,** 210–218 (1994).

219. Fitch, W. M. Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology. *Systematic Biology* **20,** 406–416 (1971).

220. Sankoff, D. Minimal Mutation Trees of Sequences. *SIAM Journal on Applied Mathematics* **28,** 35–42 (1975).

221. Day, W. H. E. & Sankoff, D. Computational Complexity of Inferring Phylogenies by Compatibility. *Systematic Biology* **35,** 224–229 (1986).

222. Lamboy, W. F. The Accuracy of the Maximum Parsimony Method for Phylogeny Reconstruction with Morphological Characters. *Systematic Botany* **19,** 489–505 (1994).

223. Tamura, K. *et al.* MEGA5: Molecular Evolutionary Genetics Analysis Using Maximum Likelihood, Evolutionary Distance, and Maximum Parsimony Methods. *Molecular Biology and Evolution* **28,** 2731–2739 (2011).

224. Xia, X. DAMBE5: A Comprehensive Software Package for Data Analysis in Molecular Biology and Evolution. *Molecular Biology and Evolution* **30,** 1720–1728 (2013).

225. Lartillot, N., Lepage, T. & Blanquart, S. PhyloBayes 3: a Bayesian software package for phylogenetic reconstruction and molecular dating. *Bioinformatics* **25,** 2286–2288 (2009).

226. Tuffley, C. & Steel, M. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. *Bulletin of Mathematical Biology* **59,** 581–607 (1997).

227. Carmel, A., Musa-Lempel, N., Tsur, D. & Ziv-Ukelson, M. The Worst Case Complexity of Maximum Parsimony. *Journal of Computational Biology* **21,** 799–808 (2014).

228. Hoang, D. T. *et al.* MPBoot: fast phylogenetic maximum parsimony tree inference and bootstrap approximation. *BMC Evolutionary Biology* **18,** 11 (2018).

229. Felsenstein, J. Maximum Likelihood and Minimum-Steps Methods for Estimating Evolutionary Trees from Data on Discrete Characters. *Systematic Biology* **22,** 240–249 (1973).

230. Yang, Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *Journal of Molecular Evolution* **39,** 306–314 (1994).

231. Kück, P., Mayer, C., Wägele, J.-W. & Misof, B. Long Branch Effects Distort Maximum Likelihood Phylogenies in Simulations Despite Selection of the Correct Model. *PLOS ONE* **7,** e36593 (2012).

232. Parks, S. L. & Goldman, N. Maximum Likelihood Inference of Small Trees in the Presence of Long Branches. *Systematic Biology* **63,** 798–811 (2014).

233. Yang, Z., Goldman, N. & Friday, A. Comparison of models for nucleotide substitution used in maximum-likelihood phylogenetic estimation. *Molecular Biology and Evolution* **11,** 316–324 (1994).

234. Abadi, S., Azouri, D., Pupko, T. & Mayrose, I. Model selection may not be a mandatory step for phylogeny reconstruction. *Nature Communications* **10,** 934 (2019).

235. Minh, B. Q. *et al.* IQ-TREE 2: New Models and Efficient Methods for Phylogenetic Inference in the Genomic Era. *Molecular Biology and Evolution* **37,** 1530–1534 (2020).

236. Nguyen, L.-T., Schmidt, H. A., von Haeseler, A. & Minh, B. Q. IQ-TREE: A Fast and Effective Stochastic Algorithm for Estimating Maximum-Likelihood Phylogenies. *Molecular Biology and Evolution* **32,** 268–274 (2015).

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

238. Yu, Y., Dong, J., Liu, K. J. & Nakhleh, L. Maximum likelihood inference of reticulate evolutionary histories. *Proceedings of the National Academy of Sciences* **111,** 16448–16453 (2014).

239. Irisarri, I. *et al.* Phylotranscriptomic consolidation of the jawed vertebrate timetree. *Nature Ecology & Evolution* **1,** 1370–1378 (2017).

240. Li, Y. *et al.* A genome-scale phylogeny of the kingdom Fungi. *Current Biology* **31,** 1653–1665.e5 (2021).

241. Zhu, Q. *et al.* Phylogenomics of 10,575 genomes reveals evolutionary proximity between domains Bacteria and Archaea. *Nature Communications* **10,** 5477 (2019).

242. Oulas, A. *et al.* Metagenomics: Tools and Insights for Analyzing Next-Generation Sequencing Data Derived from Biodiversity Studies. *Bioinformatics and Biology Insights* **9,** BBI.S12462 (2015).

243. Handelsman, J., Rondon, M. R., Brady, S. F., Clardy, J. & Goodman, R. M. Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chemistry & Biology* **5,** R245–R249 (1998).

244. Riesenfeld, C. S., Schloss, P. D. & Handelsman, J. Metagenomics: Genomic Analysis of Microbial Communities. *Annual Review of Genetics* **38,** 525–552 (2004).

245. McLaren, M. R., Willis, A. D. & Callahan, B. J. Consistent and correctable bias in metagenomic sequencing experiments. *eLife* **8,** e46923 (2019).

246. Thomas, T., Gilbert, J. & Meyer, F. Metagenomics - a guide from sampling to data analysis. *Microbial Informatics and Experimentation* **2,** 3 (2012).

247. Heather, J. M. & Chain, B. The sequence of sequencers: The history of sequencing DNA. *Genomics* **107,** 1–8 (2016).

248. Shendure, J. *et al.* DNA sequencing at 40: past, present and future. *Nature* **550,** 345–353 (2017).

249. França, L. T. C., Carrilho, E. & Kist, T. B. L. A review of DNA sequencing techniques. *Quarterly Reviews of Biophysics* **35,** 169–200 (2002).

250. Brown, T. *Gene Cloning and DNA Analysis: An Introduction* ISBN: 978-1-4443-3407-4 (John Wiley & Sons, 2010).

251. Sanger, F. & Coulson, A. R. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology* **94,** 441–448 (1975).

252. Sanger, F., Nicklen, S. & Coulson, A. R. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences* **74,** 5463–5467 (1977).

253. Sanger, F. *et al.* Nucleotide sequence of bacteriophage ΦX174 DNA. *Nature* **265,** 687–695 (1977).

254. Hunkapiller, T., Kaiser, R. J., Koop, B. F. & Hood, L. Large-scale and automated DNA sequence determination. *Science* **254,** 59–67 (1991).

255. Garrido-Cardenas, J. A., Garcia-Maroto, F., Alvarez-Bermejo, J. A. & Manzano-Agugliaro, F. DNA Sequencing Sensors: An Overview. *Sensors* **17** (2017).

256. Patel, N. *et al.* Cost analysis of standard Sanger sequencing versus next generation sequencing in the ICONIC study. *The Lancet* **388,** S86 (2016).

257. Maxam, A. M. & Gilbert, W. A new method for sequencing DNA. *Proceedings of the National Academy of Sciences* **74,** 560–564 (1977).

258. Saiki, R. K. *et al.* Primer-directed enzymatic amplification of DNA with a thermostable DNA polymerase. *Science* **239,** 487–491 (1988).

259. Riesenfeld, C. S., Schloss, P. D., Handelsman, J., *et al.* Metagenomics: genomic analysis of microbial communities. *Annual review of genetics* **38,** 525–552 (2004).

260. Koboldt, D. C., Steinberg, K. M., Larson, D. E., Wilson, R. K. & Mardis, E. R. The Next-Generation Sequencing Revolution and Its Impact on Genomics. *Cell* **155,** 27–38 (2013).

261. Nyren, P., Pettersson, B. & Uhlen, M. Solid Phase DNA Minisequencing by an Enzymatic Luminometric Inorganic Pyrophosphate Detection Assay. *Analytical Biochemistry* **208,** 171–175 (1993).

262. Ronaghi, M. Pyrosequencing Sheds Light on DNA Sequencing. *Genome Research* **11,** 3–11 (2001).

263. Margulies, M. *et al.* Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437,** 376–380 (2005).

264. Fedurco, M., Romieu, A., Williams, S., Lawrence, I. & Turcatti, G. BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies. *Nucleic Acids Research* **34,** e22–e22 (2006).

265. Bentley, D. R. *et al.* Accurate whole human genome sequencing using reversible terminator chemistry. *Nature* **456,** 53–59 (2008).

266. Quail, M. A. *et al.* A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. *BMC Genomics* **13,** 341 (2012).

267. Shendure, J. *et al.* Accurate multiplex polony sequencing of an evolved bacterial genome. *Science* **309,** 1728–1732 (2005).

268. Kircher, M. & Kelso, J. High-throughput DNA sequencing – concepts and limitations. *BioEssays* **32,** 524–536 (2010).

269. *System Specification Sheet: HiSeq® 2500 Sequencing System*

270. Tringe, S. G. & Rubin, E. M. Metagenomics: DNA sequencing of environmental samples. *Nature Reviews Genetics* **6,** 805–814 (2005).

271. Hugenholtz, P., Goebel, B. M. & Pace, N. R. Impact of Culture-Independent Studies on the Emerging Phylogenetic View of Bacterial Diversity. *Journal of Bacteriology* **180,** 4765–4774 (1998).

272. Ma, X. *et al.* Analysis of error profiles in deep next-generation sequencing data. *Genome Biology* **20,** 50 (2019).

273. Schirmer, M., D'Amore, L., Hall, N. & Quince, C. Error profiles for Next Generation sequencing technologies. *EMBnet.journal* **19,** 81–83 (2013).

274. Schirmer, M., D'Amore, R., Ijaz, U. Z., Hall, N. & Quince, C. Illumina error profiles: resolving fine-scale variation in metagenomic sequencing data. *BMC Bioinformatics* **17,** 125 (2016).

275. Eid, J. *et al.* Real-Time DNA Sequencing from Single Polymerase Molecules. *Science* **323,** 133–138 (2009).

276. Haque, F., Li, J., Wu, H.-C., Liang, X.-J. & Guo, P. Solid-state and biological nanopore for real-time sensing of single chemical and sequencing of DNA. *Nano Today* **8,** 56–74 (2013).

277. Amarasinghe, S. L. *et al.* Opportunities and challenges in long-read sequencing data analysis. *Genome Biology* **21,** 30 (2020).

278. Eisenstein, M. Oxford Nanopore announcement sets sequencing sector abuzz. *Nature Biotechnology* **30,** 295–296 (2012).

279. Jain, M., Olsen, H. E., Paten, B. & Akeson, M. The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. *Genome Biology* **17,** 239 (2016).

280. Clarke, J. *et al.* Continuous base identification for single-molecule nanopore DNA sequencing. *Nature Nanotechnology* **4,** 265–270 (2009).

281. Feng, Y., Zhang, Y., Ying, C., Wang, D. & Du, C. Nanopore-based Fourth-generation DNA Sequencing Technology. *Genomics, Proteomics & Bioinformatics* **13,** 4–16 (2015).

282. Bowden, R. *et al.* Sequencing of human genomes with nanopore technology. *Nature Communications* **10,** 1869 (2019).

283. Marchesi, J. R. & Ravel, J. The vocabulary of microbiome research: a proposal. *Microbiome* **3,** 31 (2015).

284. Bolger, A. M., Lohse, M. & Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* **30,** 2114–2120 (2014).

285. Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal* **17,** 10–12 (2011).

286. Andrews, S. *FastQC: A Quality Control Tool for High Throughput Sequence Data* 2010.

287. Edgar, R. C. & Flyvbjerg, H. Error filtering, pair assembly and error correction for next-generation sequencing reads. *Bioinformatics* **31,** 3476–3482 (2015).

288. Mande, S. S., Mohammed, M. H. & Ghosh, T. S. Classification of metagenomic sequences: methods and challenges. *Briefings in Bioinformatics* **13,** 669–681 (2012).

289. McHardy, A. C. & Rigoutsos, I. What's in the mix: phylogenetic classification of metagenome sequence samples. *Current Opinion in Microbiology. Antimicrobials/Genomics* **10,** 499–503 (2007).

290. Medvedev, P., Georgiou, K., Myers, G. & Brudno, M. *Computability of Models for Sequence Assembly* in *Algorithms in Bioinformatics* (eds Giancarlo, R. & Hannenhalli, S.) (Springer, 2007), 289–301. ISBN: 978-3-540-74126-8.

291. Ghurye, J. S., Cepeda-Espinoza, V. & Pop, M. Metagenomic Assembly: Overview, Challenges and Applications. *The Yale Journal of Biology and Medicine* **89,** 353–362 (2016).

292. Nurk, S., Meleshko, D., Korobeynikov, A. & Pevzner, P. A. metaSPAdes: a new versatile metagenomic assembler. *Genome Research* **27,** 824–834 (2017).

293. Bankevich, A. *et al.* SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *Journal of Computational Biology* **19,** 455–477 (2012).

294. Souvorov, A., Agarwala, R. & Lipman, D. J. SKESA: strategic k-mer extension for scrupulous assemblies. *Genome Biology* **19,** 153 (2018).

295. Pollard, M. O., Gurdasani, D., Mentzer, A. J., Porter, T. & Sandhu, M. S. Long reads: their purpose and place. *Human Molecular Genetics* **27,** R234–R241 (2018).

296. Antipov, D., Korobeynikov, A., McLean, J. S. & Pevzner, P. A. hybridSPAdes: an algorithm for hybrid assembly of short and long reads. *Bioinformatics* **32,** 1009–1015 (2016).

297. Vollmers, J., Wiegand, S. & Kaster, A.-K. Comparing and Evaluating Metagenome Assembly Tools from a Microbiologist's Perspective - Not Only Size Matters! *PLOS ONE* **12,** e0169662 (2017).

298. Ghosh, A., Mehta, A. & Khan, A. M. *Metagenomic Analysis and its Applications* in *Encyclopedia of Bioinformatics and Computational Biology* (eds Ranganathan, S., Gribskov, M., Nakai, K. & Schönbach, C.) (Academic Press, 2019), 184–193. ISBN: 978-0-12-811432-2.

299. Petersen, T. N. *et al.* MGmapper: Reference based mapping and taxonomy annotation of metagenomics sequence reads. *PLOS ONE* **12,** e0176469 (2017).

300. Levy Karin, E., Mirdita, M. & Söding, J. MetaEuk—sensitive, high-throughput gene discovery, and annotation for large-scale eukaryotic metagenomics. *Microbiome* **8,** 48 (2020).

301. Hoff, K. J., Lingner, T., Meinicke, P. & Tech, M. Orphelia: predicting genes in metagenomic sequencing reads. *Nucleic Acids Research* **37,** W101–W105 (2009).

302. Treangen, T. J. *et al.* MetAMOS: a modular and open source metagenomic assembly and analysis pipeline. *Genome Biology* **14,** R2 (2013).

303. Schloss, P. D. *et al.* Introducing mothur: Open-Source, Platform-Independent, Community-Supported Software for Describing and Comparing Microbial Communities. *Applied and Environmental Microbiology* **75,** 7537–7541 (2009).

304. Schloss, P. D. Reintroducing mothur: 10 Years Later. *Applied and Environmental Microbiology* **86** (2020).

305. Bolyen, E. *et al.* Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nature Biotechnology* **37,** 852–857 (2019).

306. Callahan, B. J. *et al.* DADA2: High-resolution sample inference from Illumina amplicon data. *Nature Methods* **13,** 581–583 (2016).

307. Edgar, R. C. UNOISE2: improved error-correction for Illumina 16S and ITS amplicon sequencing. *bioRxiv,* 081257 (2016).

308. Amir, A. *et al.* Deblur Rapidly Resolves Single-Nucleotide Community Sequence Patterns. *mSystems* **2** (2017).

309. Tamames, J., Cobo-Simón, M. & Puente-Sánchez, F. Assessing the performance of different approaches for functional and taxonomic annotation of metagenomes. *BMC Genomics* **20,** 960 (2019).

310. Mering, C. v. *et al.* Quantitative Phylogenetic Assessment of Microbial Communities in Diverse Environments. *Science* **315,** 1126–1130 (2007).

311. Aßhauer, K. P., Wemheuer, B., Daniel, R. & Meinicke, P. Tax4Fun: predicting functional profiles from metagenomic 16S rRNA data. *Bioinformatics* **31,** 2882–2884 (2015).

312. Sedlar, K., Kupkova, K. & Provaznik, I. Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics. *Computational and Structural Biotechnology Journal* **15,** 48–55 (2017).

313. Santamaria, M. *et al.* Reference databases for taxonomic assignment in metagenomics. *Briefings in Bioinformatics* **13,** 682–695 (2012).

314. O'Leary, N. A. *et al.* Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Research* **44,** D733–745 (2016).

315. Huson, D. H., Auch, A. F., Qi, J. & Schuster, S. C. MEGAN analysis of metagenomic data. *Genome Research* **17,** 377–386 (2007).

316. Huson, D. H. *et al.* MEGAN Community Edition - Interactive Exploration and Analysis of Large-Scale Microbiome Sequencing Data. *PLOS Computational Biology* **12,** e1004957 (2016).

317. Huson, D. H. *et al.* MEGAN-LR: new algorithms allow accurate binning and easy interactive exploration of metagenomic long reads and contigs. *Biology Direct* **13,** 6 (2018).

318. Brady, A. & Salzberg, S. L. Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. *Nature Methods* **6,** 673–676 (2009).

319. Buchfink, B., Xie, C. & Huson, D. H. Fast and sensitive protein alignment using DIAMOND. *Nature Methods* **12,** 59–60 (2015).

320. Dilthey, A. T., Jain, C., Koren, S. & Phillippy, A. M. Strain-level metagenomic assignment and compositional estimation for long reads with MetaMaps. *Nature Communications* **10,** 3066 (2019).

321. Roberts, M., Hayes, W., Hunt, B. R., Mount, S. M. & Yorke, J. A. Reducing storage requirements for biological sequence comparison. *Bioinformatics* **20,** 3363–3369 (2004).

322. Wood, D. E., Lu, J. & Langmead, B. Improved metagenomic analysis with Kraken 2. *Genome Biology* **20,** 257 (2019).

323. Breitwieser, F. P., Baker, D. N. & Salzberg, S. L. KrakenUniq: confident and fast metagenomics classification using unique k-mer counts. *Genome Biology* **19,** 198 (2018).

324. Steinegger, M. & Söding, J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology* **35,** 1026–1028 (2017).

325. Lu, J., Breitwieser, F. P., Thielen, P. & Salzberg, S. L. Bracken: estimating species abundance in metagenomics data. *PeerJ Computer Science* **3,** e104 (2017).

326. Schaeffer, L., Pimentel, H., Bray, N., Melsted, P. & Pachter, L. Pseudoalignment for metagenomic read assignment. *Bioinformatics (Oxford, England)* **33,** 2082–2088 (2017).

327. Ye, S. H., Siddle, K. J., Park, D. J. & Sabeti, P. C. Benchmarking Metagenomics Tools for Taxonomic Classification. *Cell* **178,** 779–794 (2019).

328. McIntyre, A. B. R. *et al.* Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. *Genome Biology* **18,** 182 (2017).

329. Matsen, F. A., Hoffman, N. G., Gallagher, A. & Stamatakis, A. A Format for Phylogenetic Placements. *PLOS ONE* **7,** e31009 (2012).

330. Koning, E., Phillips, M. & Warnow, T. *pplacerDC: a new scalable phylogenetic placement method* in *Proceedings of the 12th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics* (Association for Computing Machinery, 2021), 1–9. ISBN: 978-1-4503-8450-6.

331. Brown, D. G. & Truszkowski, J. LSHPlace: fast phylogenetic placement using locality-sensitive hashing. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing,* 310–319 (2013).

332. Balaban, M., Jiang, Y., Roush, D., Zhu, Q. & Mirarab, S. Fast and accurate distance-based phylogenetic placement using divide and conquer. *Molecular Ecology Resources* **22,** 1213–1227 (2022).

333. Czech, L., Barbera, P. & Stamatakis, A. Methods for automatic reference trees and multilevel phylogenetic placement. *Bioinformatics* **35,** 1151–1158 (2019).

334. Czech, L., Stamatakis, A., Dunthorn, M. & Barbera, P. Metagenomic Analysis using Phylogenetic Placement – A Review of the First Decade (2022).

335. Mirarab, S., Nguyen, N. & Warnow, T. *SEPP: SATé-Enabled Phylogenetic Placement* in *Biocomputing 2012* (WORLD SCIENTIFIC, 2011), 247–258. ISBN: 978-981-4596-37-4.

336. Berger, S. A. & Stamatakis, A. PaPaRa 2.0: a vectorized algorithm for probabilistic phylogeny-aware alignment extension. *Heidelberg Institute for Theoretical Studies* **12** (2012).

337. Barbera, P., Czech, L., Lutteropp, S. & Stamatakis, A. SCRAPP: A tool to assess the diversity of microbial samples from phylogenetic placements. *Molecular Ecology Resources* **21,** 340–349 (2021).

338. Parks, D. H., Imelfort, M., Skennerton, C. T., Hugenholtz, P. & Tyson, G. W. CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome Research* **25,** 1043–1055 (2015).

339. Douglas, G. M. *et al.* PICRUSt2 for prediction of metagenome functions. *Nature Biotechnology* **38,** 685–688 (2020).

340. Linard, B., Romashchenko, N., Pardi, F. & Rivals, E. PEWO: a collection of workflows to benchmark phylogenetic placement. *Bioinformatics* **36,** 5264–5266 (2020).

341. Huang, W., Li, L., Myers, J. R. & Marth, G. T. ART: a next-generation sequencing read simulator. *Bioinformatics (Oxford, England)* **28,** 593–594 (2012).

342. Richter, D. C., Ott, F., Auch, A. F., Schmid, R. & Huson, D. H. MetaSim—A Sequencing Simulator for Genomics and Metagenomics. *PLOS ONE* **3,** e3373 (2008).

343. Yang, C., Chu, J., Warren, R. L. & Birol, I. NanoSim: nanopore sequence read simulator based on statistical characterization. *GigaScience* **6,** 1–6 (2017).

344. Zielezinski, A. *et al.* Benchmarking of alignment-free sequence comparison methods. *Genome Biology* **20,** 144 (2019).

345. Chiaromonte, F., Yap, V. B. & Miller, W. *Scoring Pairwise Genomic Sequence Alignments.* in *Pacific Symposium on Biocomputing* (eds Altman, R. B., Dunker, A. K., Hunter, L. & Klein, T. E.) (2002), 115–126.

346. Jukes, T. H. & Cantor, C. R. *Evolution of Protein Molecules* (ed Munro, H. N.) (Academy Press, 1969).

347. Price, M. N., Dehal, P. S. & Arkin, A. P. FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments. *PLOS ONE* **5,** e9490 (2010).

348. Mölder, F. *et al. Sustainable data analysis with Snakemake* tech. rep. 10:33 (F1000Research, 2021).

349. Mahé, F. *et al.* Parasites dominate hyperdiverse soil protist communities in Neotropical rainforests. *Nature Ecology & Evolution* **1,** 1–8 (2017).

350. Sunagawa, S. *et al.* Structure and function of the global ocean microbiome. *Science* **348,** 1261359 (2015).

351. Srinivasan, S. *et al.* Bacterial Communities in Women with Bacterial Vaginosis: High Resolution Phylogenetic Analyses Reveal Relationships of Microbiota to Clinical Criteria. *PLOS ONE* **7,** e37818 (2012).

352. Kim, H.-M. *et al.* Comparative analysis of 7 short-read sequencing platforms using the Korean Reference Genome: MGI and Illumina sequencing benchmark for whole-genome sequencing. *GigaScience* **10,** giab014 (2021).

353. *Simulation of reads with ART by matthiasblanke · Pull Request #8 · phylo42/PEWO* `https://github.com/phylo42/PEWO/pull/8` (2022).

354. Howe, A. & Chain, P. S. G. Challenges and opportunities in understanding microbial communities with metagenome assembly (accompanied by IPython Notebook tutorial). *Frontiers in Microbiology* **6** (2015).

355. Parks, D. H. *et al.* Recovery of nearly 8,000 metagenome-assembled genomes substantially expands the tree of life. *Nature Microbiology* **2,** 1533–1542 (2017).

356. Kumar, K. R., Cowley, M. J. & Davis, R. L. Next-Generation Sequencing and Emerging Technologies. *Seminars in Thrombosis and Hemostasis* **45,** 661–673 (2019).

357. Balaban, M. & Mirarab, S. Phylogenetic double placement of mixed samples. *Bioinformatics* **36,** i335–i343 (2020).

358. Czech, L., Barbera, P. & Stamatakis, A. Genesis and Gappa: processing, analyzing and visualizing phylogenetic (placement) data. *Bioinformatics* **36,** 3263–3265 (2020).

359. Blanke, M. *App-SpaM* `https://github.com/matthiasblanke/App-SpaM`.

360. Blanke, M. *App-SpaM Bioconda Package* `https://anaconda.org/bioconda/appspam`.

361. Darling, A. E. *et al.* PhyloSift: phylogenetic analysis of genomes and metagenomes. *PeerJ* **2,** e243 (2014).

362. Sim, K. L. & Creamer, T. P. Abundance and Distributions of Eukaryote Protein Simple Sequences*. *Molecular & Cellular Proteomics* **1,** 983–995 (2002).

363. Marcotte, E. M., Pellegrini, M., Yeates, T. O. & Eisenberg, D. A census of protein repeats11Edited by J. M. Thornton. *Journal of Molecular Biology* **293,** 151–160 (1999).

364. Smit, A. & Green, R. H. P. RepeatMasker Open-4.0. *http://www.repeatmasker.org* (2015).

365. Morgulis, A., Gertz, E. M., Schäffer, A. A. & Agarwala, R. A Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences. *Journal of Computational Biology* **13,** 1028–1040 (2006).

366. Ames, S. K. *et al.* Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics* **29,** 2253–2260 (2013).

367. Leimeister, C.-A., Boden, M., Horwege, S., Lindner, S. & Morgenstern, B. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics* **30,** 1991–1999 (2014).

368. Levene, H. Robust tests for equality of variances, p 278–292. *Contributions to probability and statistics: essays in honor of Harold Hotelling* (1960).

369. Potter, S. C. *et al.* HMMER web server: 2018 update. *Nucleic Acids Research* **46,** W200–W204 (2018).

370. Elfmann, C. *Iterative Hash-based Sampling of Filtered Spaced-Word Matches* Report (University of Göttingen, 2021).

371. Brown, W. M., Prager, E. M., Wang, A. & Wilson, A. C. Mitochondrial DNA sequences of primates: Tempo and mode of evolution. *Journal of Molecular Evolution* **18,** 225–239 (1982).

372. Morgenstern, B. *Sequence Comparison Without Alignment: The SpaM Approaches* in *Multiple Sequence Alignment: Methods and Protocols* (ed Katoh, K.) (Springer US, 2021), 121–134. ISBN: 978-1-07-161036-7.

373. Pătraşcu, M. & Thorup, M. The Power of Simple Tabulation Hashing. *Journal of the ACM* **59,** 14:1–14:50 (2012).

374. Katz, L. S. *et al.* Mashtree: a rapid comparison of whole genome sequence files. *Journal of Open Source Software* **4,** 1762 (2019).

375. Marçais, G., Solomon, B., Patro, R. & Kingsford, C. Sketching and Sublinear Data Structures in Genomics. *Annual Review of Biomedical Data Science* **2,** 93–118 (2019).

376. LaPierre, N., Alser, M., Eskin, E., Koslicki, D. & Mangul, S. Metalign: efficient alignment-based metagenomic profiling via containment min hash. *Genome Biology* **21,** 242 (2020).

377. Katz, K. S. *et al.* STAT: a fast, scalable, MinHash-based k-mer tool to assess Sequence Read Archive next-generation sequence submissions. *Genome Biology* **22,** 270 (2021).

378. Baharav, T. Z., Kamath, G. M., Tse, D. N. & Shomorony, I. Spectral Jaccard Similarity: A New Approach to Estimating Pairwise Sequence Alignments. *Patterns* **1,** 100081 (2020).

379. Peterson, W. W. & Brown, D. T. Cyclic Codes for Error Detection. *Proceedings of the IRE* **49,** 228–235 (1961).

380. Wendte, S. *Spaced Phylo-k-mers for Phylogenetic Placement* Bachelor's Thesis (University of Göttingen, 2020).

381. Hasan, N. B., Balaban, M., Biswas, A., Bayzid, M. S. & Mirarab, S. Distance-Based Phylogenetic Placement with Statistical Support. *Biology* **11,** 1212 (2022).

382. Czech, L. & Stamatakis, A. Scalable methods for analyzing and visualizing phylogenetic placement of metagenomic samples. *PLOS ONE* **14,** e0217050 (2019).

383. Thompson, L. R. *et al.* A communal catalogue reveals Earth's multiscale microbial diversity. *Nature* **551,** 457–463 (2017).

384. Janssen, S. *et al.* Phylogenetic Placement of Exact Amplicon Sequences Improves Associations with Clinical Information. *mSystems* **3,** e00021–18 (2018).

385. Li, Y., Sidore, C., Kang, H. M., Boehnke, M. & Abecasis, G. R. Low-coverage sequencing: implications for design of complex trait association studies. *Genome Research* **21,** 940–951 (2011).

386. Li, C., Ortí, G. & Zhao, J. The phylogenetic placement of sinipercid fishes ("Perciformes") revealed by 11 nuclear loci. *Molecular Phylogenetics and Evolution* **56,** 1096–1104 (2010).

387. Spatafora, J. W., Owensby, C. A., Douhan, G. W., Boehm, E. W. & Schoch, C. L. Phylogenetic placement of the ectomycorrhizal genus Cenococcum in Gloniaceae (Dothideomycetes). *Mycologia* **104,** 758–765 (2012).

388. Mapook, A. *et al.* Taxonomic and phylogenetic placement of Nodulosphaeria. *Mycological Progress* **15,** 34 (2016).

389. Guo, X., Selden, P. A. & Ren, D. New specimens from Mid-Cretaceous Myanmar amber illuminate the phylogenetic placement of Lagonomegopidae (Arachnida: Araneae). *Zoological Journal of the Linnean Society* **195,** 399–416 (2022).

390. Kramina, T. E. *et al.* Phylogenetic Placement and Phylogeography of Large-Flowered Lotus Species (Leguminosae) Formerly Classified in Dorycnium: Evidence of Pre-Pleistocene Differentiation of Western and Eastern Intraspecific Groups. *Plants* **10,** 260 (2021).

391. Yang, Z. & Rannala, B. Molecular phylogenetics: principles and practice. *Nature Reviews Genetics* **13,** 303–314 (2012).

392. Morrison, D. A. Phylogenetic Networks: A Review of Methods to Display Evolutionary History. *Annual Research & Review in Biology,* 1518–1543 (2014).

393. Soltis, D. E. & Soltis, P. S. *Choosing an Approach and an Appropriate Gene for Phylogenetic Analysis* in *Molecular Systematics of Plants II: DNA Sequencing* (eds Soltis, D. E., Soltis, P. S. & Doyle, J. J.) (Springer US, 1998), 1–42. ISBN: 978-1-4615-5419-6.

394. Wang, Z. & Wu, M. A Phylum-Level Bacterial Phylogenetic Marker Database. *Molecular Biology and Evolution* **30,** 1258–1262 (2013).

395. Cruickshank, R. H. Molecular markers for the phylogenetics of mites and ticks. *Systematic and Applied Acarology* **7,** 3–14 (2002).

396. Philippe, H., Delsuc, F., Brinkmann, H. & Lartillot, N. Phylogenomics. *Annual Review of Ecology, Evolution, and Systematics* **36,** 541–562 (2005).

397. Bertels, F., Silander, O. K., Pachkov, M., Rainey, P. B. & van Nimwegen, E. Automated Reconstruction of Whole-Genome Phylogenies from Short-Sequence Reads. *Molecular Biology and Evolution* **31,** 1077–1088 (2014).

398. Henz, S. R., Huson, D. H., Auch, A. F., Nieselt-Struwe, K. & Schuster, S. C. Whole-genome prokaryotic phylogeny. *Bioinformatics* **21,** 2329–2335 (2005).

399. Wang, A. & Ash, G. J. Whole Genome Phylogeny of Bacillus by Feature Frequency Profiles (FFP). *Scientific Reports* **5,** 13644 (2015).

400. Rodríguez-Ezpeleta, N. *et al.* Detecting and Overcoming Systematic Errors in Genome-Scale Phylogenies. *Systematic Biology* **56,** 389–399 (2007).

401. Delsuc, F., Brinkmann, H. & Philippe, H. Phylogenomics and the reconstruction of the tree of life. *Nature Reviews Genetics* **6,** 361–375 (2005).

402. Young, A. D. & Gillung, J. P. Phylogenomics — principles, opportunities and pitfalls of big-data phylogenetics. *Systematic Entomology* **45,** 225–247 (2020).

403. De Queiroz, A. & Gatesy, J. The supermatrix approach to systematics. *Trends in Ecology & Evolution* **22,** 34–41 (2007).

404. Philippe, H. *et al.* Pitfalls in supermatrix phylogenomics. *European Journal of Taxonomy* (2017).

405. De Vienne, D. M., Ollier, S. & Aguileta, G. Phylo-MCOA: A Fast and Efficient Method to Detect Outlier Genes and Species in Phylogenomics Using Multiple Co-inertia Analysis. *Molecular Biology and Evolution* **29,** 1587–1598 (2012).

406. Walker, J. F., Brown, J. W. & Smith, S. A. Analyzing Contentious Relationships and Outlier Genes in Phylogenomics. *Systematic Biology* **67,** 916–924 (2018).

407. Izquierdo-Carrasco, F., Gagneur, J. & Stamatakis, A. Trading memory for running time in phylogenetic likelihood computations. *Heidelberg Institute for Theoretical Studies* (2011).

408. Warnow, T. Supertree Construction: Opportunities and Challenges. *arXiv* (2018).

409. Zhang, C., Rabiee, M., Sayyari, E. & Mirarab, S. ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics* **19,** 153 (2018).

410. Kay, G. L. *et al.* Eighteenth-century genomes show that mixed infections were common at time of peak tuberculosis in Europe. *Nature Communications* **6,** 6717 (2015).

411. Rabiee, M. & Mirarab, S. INSTRAL: Discordance-Aware Phylogenetic Placement Using Quartet Scores. *Systematic Biology* **69,** 384–391 (2020).

412. Jiang, Y., Balaban, M., Zhu, Q. & Mirarab, S. DEPP: Deep Learning Enables Extending Species Trees using Single Genes. *Systematic Biology,* syac031 (2022).

413. Jiang, Y., Tabaghi, P. & Mirarab, S. *Phylogenetic Placement Problem: A Hyperbolic Embedding Approach* in *Comparative Genomics* (eds Jin, L. & Durand, D.) (Springer International Publishing, 2022), 68–85. ISBN: 978-3-031-06220-9.

414. Zaharias, P. & Warnow, T. Recent progress on methods for estimating and updating large phylogenies. *Philosophical Transactions of the Royal Society B: Biological Sciences* **377,** 20210244 (2022).

415. Dalquen, D. A., Anisimova, M., Gonnet, G. H. & Dessimoz, C. ALF—A Simulation Framework for Genome Evolution. *Molecular Biology and Evolution* **29,** 1115–1123 (2012).

416. Zhang, Z. & Gerstein, M. Patterns of nucleotide substitution, insertion and deletion in the human genome inferred from pseudogenes. *Nucleic Acids Research* **31,** 5338–5348 (2003).

417. Benner, S. A., Cohen, M. A. & Gonnet, G. H. Empirical and Structural Models for Insertions and Deletions in the Divergent Evolution of Proteins. *Journal of Molecular Biology* **229,** 1065–1082 (1993).

418. Robinson, D. F. & Foulds, L. Comparison of phylogenetic trees. *Mathematical Biosciences* **53,** 131–147 (1981).

419. Vachaspati, P. & Warnow, T. FastRFS: fast and accurate Robinson-Foulds Supertrees using constrained exact optimization. *Bioinformatics* **33,** 631–639 (2017).

420. Smith, M. R. Information theoretic generalized Robinson–Foulds metrics for comparing phylogenetic trees. *Bioinformatics* **36,** 5007–5013 (2020).

421. Bogdanowicz, D. & Giaro, K. On a matching distance between rooted phylogenetic trees. *International Journal of Applied Mathematics and Computer Science,* 669–684 (2013).

422. Bogdanowicz, D. & Giaro, K. Matching Split Distance for Unrooted Binary Phylogenetic Trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **9,** 150–160 (2012).

423. Llabrés, M., Rosselló, F. & Valiente, G. *A Generalized Robinson-Foulds Distance for Clonal Trees, Mutation Trees, and Phylogenetic Trees and Networks* in *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics* (Association for Computing Machinery, 2020), 1–10. ISBN: 978-1-4503-7964-9.

424. Mallo, D., De Oliveira Martins, L. & Posada, D. SimPhy: Phylogenomic Simulation of Gene, Locus, and Species Trees. *Systematic Biology* **65,** 334–344 (2016).

425. Gogarten, J. P. & Townsend, J. P. Horizontal gene transfer, genome innovation and evolution. *Nature Reviews Microbiology* **3,** 679–687 (2005).

426. Degnan, J. H. & Rosenberg, N. A. Gene tree discordance, phylogenetic inference and the multispecies coalescent. *Trends in Ecology & Evolution* **24,** 332–340 (2009).

427. Posada, D. & Crandall, K. A. The Effect of Recombination on the Accuracy of Phylogeny Estimation. *Journal of Molecular Evolution* **54,** 396–402 (2002).

428. Walker, J. F., Brown, J. W. & Smith, S. A. Analyzing Contentious Relationships and Outlier Genes in Phylogenomics. *Systematic Biology* **67,** 916–924 (2018).

429. Römer, P.-F. *Gene outlier detection using alignment-free methods* Bachelor's Thesis (University of Göttingen, 2022).

430. Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. *A density-based algorithm for discovering clusters in large spatial databases with noise.* in *kdd* **96** (1996), 226–231.

431. Hillis, D. M., Heath, T. A. & St. John, K. Analysis and Visualization of Tree Space. *Systematic Biology* **54,** 471–482 (2005).

432. Smith, M. R. Robust Analysis of Phylogenetic Tree Space. *Systematic Biology* **71,** 1255–1270 (2022).

433. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).

434. Boukerche, A., Zheng, L. & Alfandi, O. Outlier Detection: Methods, Models, and Classification. *ACM Computing Surveys* **53,** 55:1–55:37 (2020).

435. Neumann, U. *et al.* Compensation of feature selection biases accompanied with improved predictive performance for binary classification by using a novel ensemble feature selection approach. *BioData Mining* **9,** 36 (2016).

436. Regendantz, R. F. D. *Using Read-SpaM to estimate phylogenetic distances between primates from ancient DNA short reads* Bachelor's Thesis (University of Göttingen, 2022).

437. Birth, N., Dencker, T. & Morgenstern, B. Insertions and deletions as phylogenetic signal in an alignment-free context. *PLOS Computational Biology* **18,** e1010303 (2022).

438. Sessions, S. K. *Genome Size* in *Brenner's Encyclopedia of Genetics (Second Edition)* (eds Maloy, S. & Hughes, K.) (Academic Press, 2013), 301–305. ISBN: 978-0-08-096156-9.

439. Piovesan, A. *et al.* Human protein-coding genes and gene feature statistics in 2019. *BMC Research Notes* **12,** 315 (2019).

440. Haubold, B. & Wiehe, T. How repetitive are genomes? *BMC Bioinformatics* **7,** 541 (2006).

441. Koning, A. P. J. d., Gu, W., Castoe, T. A., Batzer, M. A. & Pollock, D. D. Repetitive Elements May Comprise Over Two-Thirds of the Human Genome. *PLOS Genetics* **7,** e1002384 (2011).

442. Smith, N. G. C., Webster, M. T. & Ellegren, H. Deterministic Mutation Rate Variation in the Human Genome. *Genome Research* **12,** 1350–1356 (2002).

443. Fan, H., Ives, A. R., Surget-Groba, Y. & Cannon, C. H. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC Genomics* **16,** 522 (2015).

444. Sarmashghi, S., Bohmann, K., P. Gilbert, M. T., Bafna, V. & Mirarab, S. Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome Biology* **20,** 34 (2019).

445. Kent, W. J. *et al.* The Human Genome Browser at UCSC. *Genome Research* **12,** 996–1006 (2002).

446. Sims, G. E., Jun, S.-R., Wu, G. A. & Kim, S.-H. Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences* **106,** 2677–2682 (2009).

447. Ren, J. *et al.* Alignment-Free Sequence Analysis and Applications. *Annual review of biomedical data science* **1,** 93–114 (2018).

448. Delibaş, E., Arslan, A., Şeker, A. & Diri, B. A novel alignment-free DNA sequence similarity analysis approach based on top-k n-gram match-up. *Journal of Molecular Graphics and Modelling* **100,** 107693 (2020).

449. Deorowicz, S., Kokot, M., Grabowski, S. & Debudaj-Grabysz, A. KMC 2: fast and resource-frugal k-mer counting. *Bioinformatics* **31,** 1569–1576 (2015).

450. Rizk, G., Lavenier, D. & Chikhi, R. DSK: k-mer counting with very low memory usage. *Bioinformatics* **29,** 652–653 (2013).

451. Babenko, M. A. & Starikovskaya, T. A. *Computing Longest Common Substrings Via Suffix Arrays* in *Computer Science - Theory and Applications* **5010** (Springer, 2008), 64–75.

452. Morgenstern, B. *Sequence Comparison without Alignment: The SpaM approaches* in *Multiple Sequence Alignment* (ed Katoh, K.) (Springer, 2020).

453. Gawrychowski, P., Kosche, M., Koß, T., Manea, F. & Siemer, S. Efficiently Testing Simon's Congruence. *arXiv,* 18 (2021).

454. Pace, N. R., Sapp, J. & Goldenfeld, N. Phylogeny and beyond: Scientific, historical, and conceptual significance of the first tree of life. *Proceedings of the National Academy of Sciences* **109,** 1011–1018 (2012).

455. Bowern, C. Computational Phylogenetics. *Annual Review of Linguistics* **4,** 281–296 (2018).

456. Retzlaff, N. & Stadler, P. F. Phylogenetics beyond biology. *Theory in Biosciences* **137,** 133–143 (2018).

457. Darwin, C. R. *Notebook B: Transmutation of species* (Darwin Online, 1837).

458. Nakhleh, L. *Evolutionary Phylogenetic Networks: Models and Issues* in *Problem Solving Handbook in Computational Biology and Bioinformatics* (eds Heath, L. S. & Ramakrishnan, N.) (Springer US, 2011), 125–158. ISBN: 978-0-387-09760-2.

459. Huson, D. H., Rupp, R. & Scornavacca, C. *Phylogenetic Networks: Concepts, Algorithms and Applications* ISBN: 978-1-139-49287-4 (Cambridge University Press, 2010).

460. Syvanen, M. Evolutionary Implications of Horizontal Gene Transfer. *Annual Review of Genetics* **46,** 341–358 (2012).

461. Ward, L. M., Hemp, J., Shih, P. M., McGlynn, S. E. & Fischer, W. W. Evolution of Phototrophy in the Chloroflexi Phylum Driven by Horizontal Gene Transfer. *Frontiers in Microbiology* **9** (2018).

462. Kurland, C. G., Canback, B. & Berg, O. G. Horizontal gene transfer: A critical view. *Proceedings of the National Academy of Sciences* **100,** 9658–9662 (2003).

463. Richardson, A. O. & Palmer, J. D. Horizontal gene transfer in plants. *Journal of Experimental Botany* **58,** 1–9 (2007).

464. Chang, K.-Y., Cui, Y., Yiu, S.-M. & Hon, W.-K. Reconstructing One-Articulated Networks with Distance Matrices. *Journal of Computational Biology* **25,** 253–269 (2018).

465. Bordewich, M., Semple, C. & Tokac, N. Constructing Tree-Child Networks from Distance Matrices. *Algorithmica* **80,** 2240–2259 (2018).

466. Whelan, S., Liò, P. & Goldman, N. Molecular phylogenetics: state-of-the-art methods for looking into the past. *Trends in Genetics* **17,** 262–272 (2001).

467. Sullivan, J. & Joyce, P. Model Selection in Phylogenetics. *Annual Review of Ecology, Evolution, and Systematics* **36,** 445–466 (2005).

468. Nei, M., Kumar, S. & Nei, E. P. *Molecular Evolution and Phylogenetics* ISBN: 978-0-19-513585-5 (Oxford University Press, 2000).

469. Bray, T. *The javascript object notation (json) data interchange format* tech. rep. (2014).

470. Ghai, R., Mizuno, C. M., Picazo, A., Camacho, A. & Rodriguez-Valera, F. Metagenomics uncovers a new group of low GC and ultra-small marine Actinobacteria. *Scientific Reports* **3,** 2471 (2013).

471. Tully, B. J., Sachdeva, R., Graham, E. D. & Heidelberg, J. F. 290 metagenome-assembled genomes from the Mediterranean Sea: a resource for marine microbiology. *PeerJ* **5,** e3558 (2017).

472. Gaio, D. *et al.* Phylogenetic diversity analysis of shotgun metagenomic reads describes gut microbiome development and treatment effects in the post-weaned pig. *PLOS ONE* **17,** e0270372 (2022).

473. Balvočiūtė, M. & Huson, D. H. SILVA, RDP, Greengenes, NCBI and OTT — how do these taxonomies compare? *BMC Genomics* **18,** 114 (2017).

474. Hinchliff, C. E. *et al.* Synthesis of phylogeny and taxonomy into a comprehensive tree of life. *Proceedings of the National Academy of Sciences* **112,** 12764–12769 (2015).

475. Zheng, Q., Bartow-McKenney, C., Meisel, J. S. & Grice, E. A. HmmUFOtu: An HMM and phylogenetic placement based ultra-fast taxonomic assignment and OTU picking tool for microbiome amplicon sequencing studies. *Genome Biology* **19,** 82 (2018).

476. Martiniano, R., De Sanctis, B., Hallast, P. & Durbin, R. Placing Ancient DNA Sequences into Reference Phylogenies. *Molecular Biology and Evolution* **39,** msac017 (2022).

477. Filipski, A., Tamura, K., Billing-Ross, P., Murillo, O. & Kumar, S. Phylogenetic placement of metagenomic reads using the minimum evolution principle. *BMC Genomics* **16,** S13 (2015).

478. Nabhan, A. R. & Sarkar, I. N. The impact of taxon sampling on phylogenetic inference: a review of two decades of controversy. *Briefings in Bioinformatics* **13,** 122–134 (2012).

479. Hu, T., Chitnis, N., Monos, D. & Dinh, A. Next-generation sequencing technologies: An overview. *Human Immunology. Next Generation Sequencing and its Application to Medical Laboratory Immunology* **82,** 801–811 (2021).

480. Wedell, E., Cai, Y. & Warnow, T. SCAMPP: Scaling Alignment-based Phylogenetic Placement to Large Trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* 1 (2022).

481. Wedell, E., Cai, Y. & Warnow, T. *Scalable and Accurate Phylogenetic Placement Using pplacer-XR* in *Algorithms for Computational Biology* (eds Martín-Vide, C., Vega-Rodríguez, M. A. & Wheeler, T.) (Springer International Publishing, 2021), 94–105. ISBN: 978-3-030-74432-8.

482. Manni, M., Berkeley, M. R., Seppey, M., Simão, F. A. & Zdobnov, E. M. BUSCO Update: Novel and Streamlined Workflows along with Broader and Deeper Phylogenetic Coverage for Scoring of Eukaryotic, Prokaryotic, and Viral Genomes. *Molecular Biology and Evolution* **38,** 4647–4654 (2021).

483. Sahl, J. W. *et al.* Phylogenetically typing bacterial strains from partial SNP genotypes observed from direct sequencing of clinical specimen metagenomic data. *Genome Medicine* **7,** 52 (2015).

484. Turakhia, Y. *et al.* Ultrafast Sample placement on Existing tRees (UShER) enables real-time phylogenetics for the SARS-CoV-2 pandemic. *Nature Genetics* **53,** 809–816 (2021).

485. Wagner, D. L., Grames, E. M., Forister, M. L., Berenbaum, M. R. & Stopak, D. Insect decline in the Anthropocene: Death by a thousand cuts. *Proceedings of the National Academy of Sciences* **118,** e2023989118 (2021).

486. Oliver, R. Y., Meyer, C., Ranipeta, A., Winner, K. & Jetz, W. Global and national trends, gaps, and opportunities in documenting and monitoring species distributions. *PLOS Biology* **19,** e3001336 (2021).

Appendices

Chapter A

App-SpaM

App-SpaM is publicly and freely available for download on its Github repository, including a detailed documentation, at `www.github.com/matthiasblanke/App-SpaM`.

A.1 Methods

Supplementary Table A.1 contains additional information on all data sets besides those listed in Tab. 3.1. In their order of appearance, columns contain:

abbreviation	The name of the dataset.
$\frac{d_f}{d_c}$	The fraction of the distance between the root and the farthest leaf node (d_f) and the distance between the root and the closest leaf node (d_c).
$l_{\max} - l_{\min}$	The absolute difference in sequence length between the longest and shortest reference sequence.
$\frac{l_{\min}}{l_{\max}}$	The relative difference in sequence length between the longest and shortest reference sequence.

We performed a comprehensive evaluation including a large range of parameters for each program. All tested parameter combinations of all PP programs are contained in Suppl. Tab. A.2, together with a short description of each parameter. App-SpaM uses a placement heuristic, also referred to as its *mode*, to infer placement positions from spaced-word matches statistics. We tested all of the introduced modes with varying pattern weights $w \in \{8, 12, 16\}$. In the accuracy evaluation, spaced words are extracted on the basis of a single pattern. Only in dedicated experiments, multiple patterns were employed; see, for example, Suppl. Fig. A.1. The placement criteria (*crit*) of APPLES are least squares phylogenetic placement (MLSE), minimum evolution (ME), or a combination of both (HYBRID). Three least squares methods (*meth*) are implemented and were tested, see Eq. 2.22: OLS is ordinary least squares with $k = 0$, BE uses $k = 1$, and FM $k = 2$. For RAPPAS, the value k of the phylo-k-mers greatly influences its performance: for larger values of k results are more accurate but execution speed is slower. pplacer follows its 'baseball' heuristic which is based upon three parameters (ms, sb, mp). For EPA-ng we tested the newest and fastest heuristic ($h = 1$) and an older heuristic equivalent to the one employed in EPA ($h = 2$). For the older heuristic (and for EPA), a single parameter g specifies the proportion of branch lengths for which full branch length optimization is performed. In both cases, we tested $g = 0.01$ and $g = 0.1$. All notations follow the abbreviations used within the configuration files of PEWO. Parameters were chosen based on available documentations of the software packages themselves and information provided by the PEWO framework.

Table A.1 – Additional information on data sets used in the PAC and RES evaluations.

abbreviation	$\frac{d_f}{d_c}$	$l_{\max} - l_{\min}$	$\frac{l_{\min}}{l_{\max}}$
BAC-150	129.9	264	0.79
HIV-104	8.67	1781	0.81
NEOTROP-512	17.86	1895	0.35
TARA-3748	72.68	605	0.63
BV-797	341.42	1653	0.24
EPA-218	37.76	217	0.86
EPA-628	21.85	780	0.22
EPA-714	13.89	396	0.68
WOL-43	2.12	1136292	0.37
CPU-652	45.32	1055	0.34
CPU-512	17.86	1895	0.35

Table A.2 – Parameter choices for each placement program.

program	parameter	choices	description
App-SpaM			
	mode	SpaM-2, SpaM-4, Min-Dist,LCA-Dist, Min-Count, LCA-Count	placement heuristics
	w	8, 12, 16	weight of patterns
	pattern	1	number of patterns
APPLES			
	meth	BE, FM, OLS	least squares method
	crit	HYBRID, ME, MLSE	placement criterion
RAPPAS			
	k	6, 7, 8	size of phylo-k-mers
	o	1.5, 2.0	probability threshold for RAPPAS
	red	0.99	reduction: gap/non-gap ratio above which site of alignment is ignored
	ar	RAXMLNG	software for ancestral state reconstruction
PPLACER			
	ms	1, 3, 6	max-strikes
	sb	1, 3, 6	strike-box
	mp	40	max-pitches
EPA-NG			
	h	1, 2	heuristic
EPA			
	g	0.01, 0.1	proportion of top scoring branches for which full optimization is computed

A.2 Results

The subsequent tables contain detailed statistics for the box plots of the accuracy evaluation shown in Sec. 3.1.2. The rows in their order of appearance in each table contain:

w	Pattern weight for App-SpaM. Only given if applicable, otherwise the default $w = 12$ was used.
count	The number of pruning repetitions in the box plot.
mean	The average ND over all repetitions.
std	The standard deviation over all repetitions.
min	The minimal ND across all repetitions.
25	The first quartile of ND across all repetitions.
50	The second quartile of ND across all repetitions.
75	The third quartile of ND across all repetitions.
max	The maximal ND across all repetitions.

Heuristics and pattern weight

The following tables show detailed statistics for Fig. 3.3 to Fig. 3.5, which show average node distances for App-SpaM dependent on the employed heuristics for the data sets bac-150 and hiv-104.

Table A.3 – Summary statistics for the box plots shown in Fig. 3.4.

mode	w	count	mean	std	min	25	50	75	max
BESTCOUNT	8.00	100.00	8.28	1.78	3.00	7.37	8.09	9.15	15.00
BESTCOUNT	12.00	100.00	8.31	1.72	3.25	7.37	8.09	9.16	14.13
BESTCOUNT	16.00	100.00	8.27	1.70	3.63	7.30	8.12	9.10	13.56
BESTSCORE	8.00	100.00	8.71	1.88	4.19	7.55	8.54	9.94	14.31
BESTSCORE	12.00	100.00	8.98	2.08	4.00	7.69	8.57	10.20	15.75
BESTSCORE	16.00	100.00	9.01	2.06	4.03	7.69	8.69	10.12	16.31
LCACOUNT	8.00	100.00	4.78	1.92	1.00	3.59	4.39	5.63	10.63
LCACOUNT	12.00	100.00	4.73	1.79	1.00	3.66	4.41	5.62	9.75
LCACOUNT	16.00	100.00	4.82	1.83	1.25	3.63	4.53	5.70	9.84
LCASCORE	8.00	100.00	8.11	2.28	2.88	6.82	8.07	9.64	14.50
LCASCORE	12.00	100.00	8.22	2.36	2.31	6.88	8.20	9.69	14.75
LCASCORE	16.00	100.00	8.22	2.40	2.31	6.86	8.21	9.58	14.75
EXP2	8.00	100.00	4.70	1.93	1.00	3.50	4.36	5.32	10.63
EXP2	12.00	100.00	4.67	1.87	1.00	3.47	4.36	5.72	9.68
EXP2	16.00	100.00	4.76	1.94	1.00	3.59	4.50	5.88	9.70
EXP4	8.00	100.00	4.65	2.01	1.00	3.50	4.36	5.33	10.63
EXP4	12.00	100.00	4.65	1.94	1.00	3.47	4.40	5.70	9.68
EXP4	16.00	100.00	4.71	2.00	1.00	3.50	4.43	5.67	9.70
APPLES	8.00	100.00	5.90	2.32	2.19	4.20	5.59	7.24	13.63
APPLES	12.00	100.00	5.81	2.26	1.50	4.17	5.34	7.11	12.94
APPLES	16.00	100.00	5.97	2.24	1.50	4.35	5.69	7.06	12.81

Table A.4 – Summary statistics for the box plots shown in Fig. 3.3.

mode	w	count	mean	std	min	25	50	75	max
BESTCOUNT	12.00	100.00	8.31	1.72	3.25	7.37	8.09	9.16	14.13
BESTSCORE	12.00	100.00	8.98	2.08	4.00	7.69	8.57	10.20	15.75
LCACOUNT	12.00	100.00	4.73	1.79	1.00	3.66	4.41	5.62	9.75
LCASCORE	12.00	100.00	8.22	2.36	2.31	6.88	8.20	9.69	14.75
EXP4	12.00	100.00	4.65	1.94	1.00	3.47	4.40	5.70	9.68
APPLES	12.00	100.00	5.81	2.26	1.50	4.17	5.34	7.11	12.94

Table A.5 – Summary statistics for the box plots shown in Fig. 3.5a.

mode	w	count	mean	std	min	25	50	75	max
SPAMCOUNT	8.00	100.00	5.23	1.37	1.49	4.59	5.37	5.98	9.16
SPAMCOUNT	12.00	100.00	5.23	1.37	1.47	4.70	5.34	5.87	8.88
SPAMCOUNT	16.00	100.00	5.32	1.33	1.57	4.73	5.39	6.02	8.48
MINDIST	8.00	100.00	5.74	1.47	1.55	5.12	6.00	6.62	9.06
MINDIST	12.00	100.00	6.35	1.52	1.80	5.66	6.60	7.34	9.32
MINDIST	16.00	100.00	6.72	1.62	1.93	6.06	6.99	7.74	9.92
LCACOUNT	8.00	100.00	4.08	1.23	1.61	3.31	3.95	4.85	8.00
LCACOUNT	12.00	100.00	4.14	1.25	1.54	3.29	4.09	4.86	8.02
LCACOUNT	16.00	100.00	4.22	1.29	1.54	3.36	4.12	4.93	8.23
LCADIST	8.00	100.00	4.46	1.43	1.58	3.49	4.38	5.23	8.73
LCADIST	12.00	100.00	4.84	1.60	1.75	3.70	4.77	5.72	9.88
LCADIST	16.00	100.00	5.03	1.68	1.80	3.73	4.92	5.97	9.76
EXP2	8.00	100.00	4.08	1.23	1.61	3.31	3.95	4.85	8.00
EXP2	12.00	100.00	4.13	1.26	1.49	3.29	4.09	4.87	8.02
EXP2	16.00	100.00	4.21	1.31	1.37	3.33	4.12	4.94	8.25
EXP4	8.00	100.00	4.07	1.24	1.33	3.27	3.96	4.85	8.00
EXP4	12.00	100.00	4.13	1.25	1.23	3.29	4.10	4.86	7.98
EXP4	16.00	100.00	4.21	1.27	1.27	3.42	4.09	4.97	8.17

Table A.6 – Summary statistics for the box plots shown in Fig. 3.5b.

mode	w	count	mean	std	min	25	50	75	max
SPAMCOUNT	8.00	100.00	3.70	1.58	1.00	2.80	3.44	4.38	8.47
SPAMCOUNT	12.00	100.00	3.78	1.62	1.00	2.75	3.52	4.50	8.71
SPAMCOUNT	16.00	100.00	3.85	1.61	1.00	2.82	3.51	4.46	9.06
MINDIST	8.00	100.00	4.04	1.62	1.00	3.05	3.80	4.90	8.91
MINDIST	12.00	100.00	4.91	1.66	1.06	3.92	4.82	5.94	9.49
MINDIST	16.00	100.00	5.79	1.68	1.06	4.83	5.95	6.96	9.35
LCACOUNT	8.00	100.00	2.99	0.90	1.51	2.42	2.91	3.35	6.47
LCACOUNT	12.00	100.00	3.01	0.93	1.43	2.41	2.98	3.42	6.53
LCACOUNT	16.00	100.00	3.08	0.90	1.53	2.47	3.00	3.51	6.24
LCADIST	8.00	100.00	3.24	0.94	1.59	2.61	3.16	3.81	6.16
LCADIST	12.00	100.00	3.83	1.23	1.59	2.88	3.70	4.53	7.41
LCADIST	16.00	100.00	4.42	1.56	1.79	3.23	4.28	5.24	9.18
EXP2	8.00	100.00	2.98	0.91	1.39	2.42	2.91	3.35	6.47
EXP2	12.00	100.00	3.00	0.94	1.12	2.41	2.98	3.42	6.53
EXP2	16.00	100.00	3.07	0.92	1.07	2.47	3.00	3.51	6.24
EXP4	8.00	100.00	2.98	0.91	1.07	2.42	2.91	3.35	6.47
EXP4	12.00	100.00	2.99	0.95	1.01	2.38	2.97	3.42	6.53
EXP4	16.00	100.00	3.04	0.94	1.00	2.38	2.97	3.53	6.24

Size of pattern sets

The following two tables show detailed statistics for the evaluation of pattern sets of varying sizes on the BAC-150 and BV-797 data sets. The names of the columns follow the conventions introduced above in Suppl. Sec. A.2. Suppl. Fig. A.1 shows the average node distance for App-SpaM dependent on the number of patterns for the BV-797 data set.

Table A.7 – Summary statistics for the box plots shown in Fig. 3.8.

mode	w	pattern	count	mean	std	min	25	50	75	max
LCACOUNT	1.00	1.00	100.00	4.74	1.80	1.00	3.50	4.36	5.82	9.68
LCACOUNT	1.00	2.00	100.00	4.82	1.90	1.00	3.59	4.58	5.60	10.25
LCACOUNT	1.00	3.00	100.00	4.81	1.88	1.00	3.63	4.44	5.95	9.88
LCACOUNT	1.00	4.00	100.00	4.78	1.87	1.00	3.59	4.48	5.69	9.85
LCACOUNT	1.00	5.00	100.00	4.78	1.87	1.00	3.59	4.50	5.72	10.63
LCACOUNT	2.00	1.00	100.00	4.73	1.92	1.00	3.50	4.36	5.64	10.00
LCACOUNT	2.00	2.00	100.00	4.79	1.87	1.00	3.50	4.44	5.76	10.00
LCACOUNT	2.00	3.00	100.00	4.79	1.88	1.00	3.50	4.50	5.71	10.88
LCACOUNT	2.00	4.00	100.00	4.77	1.82	1.00	3.49	4.63	5.73	9.88
LCACOUNT	2.00	5.00	100.00	4.77	1.85	1.00	3.38	4.50	5.72	10.00
LCACOUNT	3.00	1.00	100.00	4.71	1.87	1.00	3.48	4.27	5.75	10.35
LCACOUNT	3.00	2.00	100.00	4.72	1.87	1.00	3.59	4.45	5.68	10.25
LCACOUNT	3.00	3.00	100.00	4.77	1.90	1.00	3.58	4.50	5.63	10.15
LCACOUNT	3.00	4.00	100.00	4.72	1.89	1.00	3.47	4.40	5.53	9.83
LCACOUNT	3.00	5.00	100.00	4.77	1.92	1.00	3.47	4.44	5.67	10.25
LCACOUNT	4.00	1.00	100.00	4.79	1.88	1.08	3.63	4.40	5.73	9.88
LCACOUNT	4.00	2.00	100.00	4.77	1.85	1.00	3.63	4.54	5.58	9.88
LCACOUNT	4.00	3.00	100.00	4.73	1.87	1.00	3.48	4.56	5.61	9.93
LCACOUNT	4.00	4.00	100.00	4.77	1.88	1.00	3.38	4.50	5.65	9.68
LCACOUNT	4.00	5.00	100.00	4.76	1.92	1.00	3.50	4.49	5.66	10.38
LCACOUNT	5.00	1.00	100.00	4.79	1.85	1.00	3.63	4.50	5.76	9.29
LCACOUNT	5.00	2.00	100.00	4.76	1.85	1.42	3.50	4.38	5.64	9.88
LCACOUNT	5.00	3.00	100.00	4.74	1.85	1.21	3.47	4.50	5.64	10.13
LCACOUNT	5.00	4.00	100.00	4.76	1.84	1.08	3.61	4.50	5.53	9.53
LCACOUNT	5.00	5.00	100.00	4.74	1.85	1.08	3.59	4.47	5.62	10.08
SPAMX	1.00	1.00	100.00	4.65	1.94	1.00	3.47	4.40	5.70	9.68
SPAMX	1.00	2.00	100.00	4.71	2.03	1.00	3.50	4.54	5.51	10.25
SPAMX	1.00	3.00	100.00	4.70	2.00	1.00	3.59	4.32	5.71	9.88
SPAMX	1.00	4.00	100.00	4.66	1.99	1.00	3.50	4.36	5.61	9.85
SPAMX	1.00	5.00	100.00	4.69	1.99	1.00	3.50	4.50	5.68	10.63
SPAMX	2.00	1.00	100.00	4.63	2.04	1.00	3.47	4.25	5.49	10.00
SPAMX	2.00	2.00	100.00	4.69	1.99	1.00	3.50	4.36	5.63	10.00
SPAMX	2.00	3.00	100.00	4.69	2.00	1.00	3.50	4.45	5.69	10.88
SPAMX	2.00	4.00	100.00	4.65	1.95	1.00	3.45	4.50	5.59	9.88
SPAMX	2.00	5.00	100.00	4.67	1.98	1.00	3.38	4.50	5.63	10.00
SPAMX	3.00	1.00	100.00	4.60	1.98	1.00	3.39	4.25	5.76	10.35
SPAMX	3.00	2.00	100.00	4.60	1.97	1.00	3.50	4.34	5.45	10.25
SPAMX	3.00	3.00	100.00	4.67	2.01	1.00	3.43	4.41	5.45	10.15
SPAMX	3.00	4.00	100.00	4.63	2.01	1.00	3.38	4.30	5.50	9.83
SPAMX	3.00	5.00	100.00	4.66	2.04	1.00	3.38	4.39	5.53	10.25
SPAMX	4.00	1.00	100.00	4.69	1.99	1.00	3.50	4.37	5.75	9.88
SPAMX	4.00	2.00	100.00	4.65	1.98	1.00	3.38	4.47	5.52	9.88
SPAMX	4.00	3.00	100.00	4.61	1.97	1.00	3.39	4.50	5.50	9.93
SPAMX	4.00	4.00	100.00	4.67	2.00	1.00	3.38	4.44	5.56	9.68
SPAMX	4.00	5.00	100.00	4.67	2.03	1.00	3.45	4.37	5.45	10.38
SPAMX	5.00	1.00	100.00	4.69	1.98	1.00	3.50	4.50	5.63	9.29
SPAMX	5.00	2.00	100.00	4.64	1.95	1.00	3.42	4.38	5.63	9.88
SPAMX	5.00	3.00	100.00	4.64	1.96	1.00	3.42	4.50	5.63	10.13
SPAMX	5.00	4.00	100.00	4.65	1.96	1.00	3.54	4.44	5.51	9.53
SPAMX	5.00	5.00	100.00	4.64	1.96	1.00	3.50	4.43	5.50	10.08

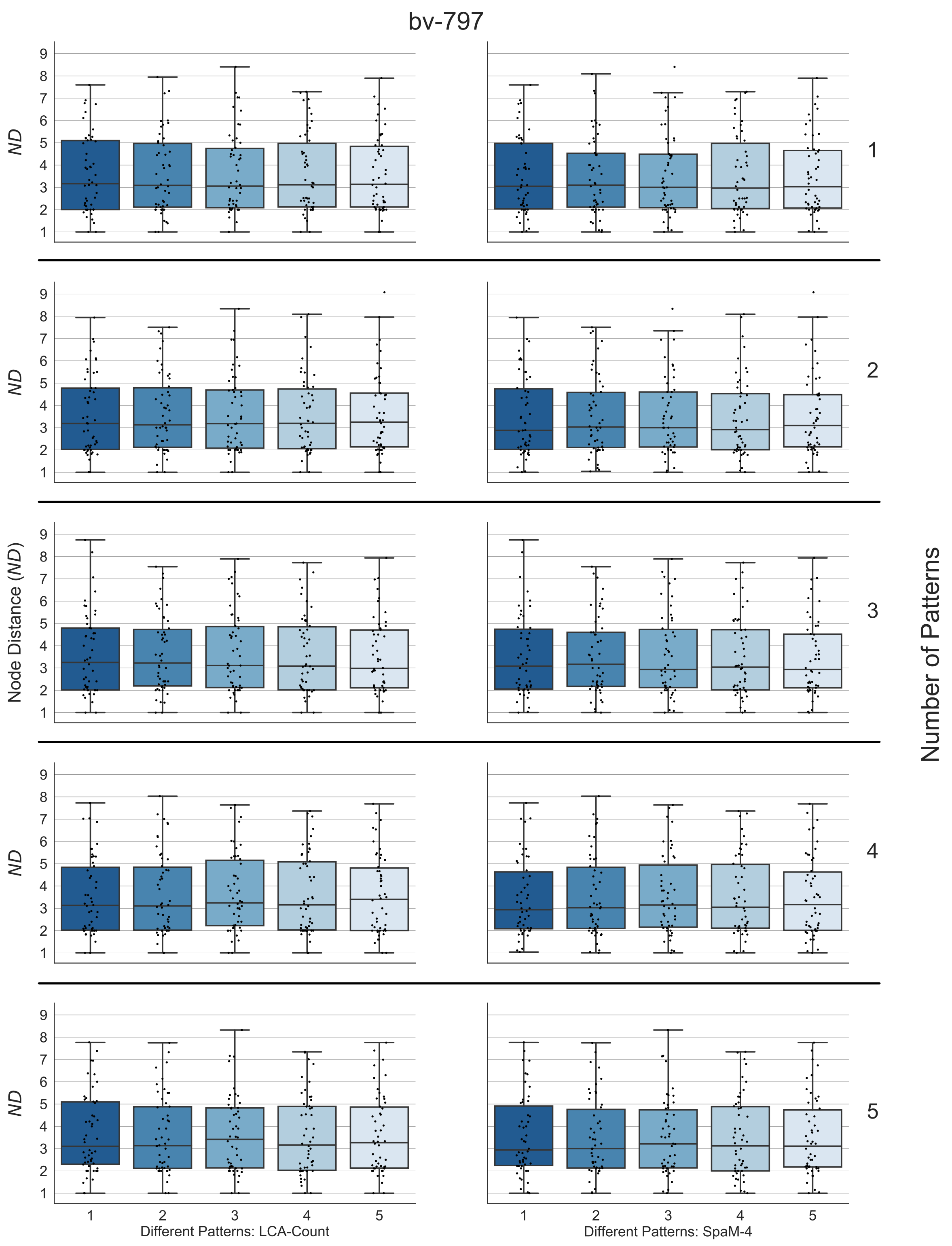


Figure A.1 – Accuracy (left y-axes) of the LCA-COUNT and SPAM-4 heuristics (outer x-axis) dependent on the number of used patterns (right y-axis). For each pattern set, five repetitions with different random seeds for RASBHARI are shown (inner x-axes, different color hues). Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

Table A.8 – Summary statistics for the box plots shown in Suppl. Fig. A.1.

mode	w	pattern	count	mean	std	min	25	50	75	max
LCACOUNT	1.00	1.00	50.00	3.57	1.75	1.00	2.00	3.17	5.10	7.60
LCACOUNT	1.00	2.00	50.00	3.59	1.78	1.00	2.03	3.19	4.78	7.94
LCACOUNT	1.00	3.00	50.00	3.62	1.86	1.00	2.01	3.25	4.79	8.75
LCACOUNT	1.00	4.00	50.00	3.52	1.74	1.00	2.02	3.13	4.84	7.73
LCACOUNT	1.00	5.00	50.00	3.66	1.78	1.00	2.30	3.11	5.10	7.77
LCACOUNT	2.00	1.00	50.00	3.58	1.77	1.00	2.11	3.09	4.97	7.95
LCACOUNT	2.00	2.00	50.00	3.64	1.76	1.00	2.12	3.13	4.79	7.51
LCACOUNT	2.00	3.00	50.00	3.63	1.74	1.00	2.19	3.22	4.73	7.55
LCACOUNT	2.00	4.00	50.00	3.61	1.84	1.00	2.02	3.11	4.85	8.03
LCACOUNT	2.00	5.00	50.00	3.57	1.73	1.00	2.11	3.13	4.88	7.75
LCACOUNT	3.00	1.00	50.00	3.63	1.83	1.00	2.08	3.06	4.75	8.40
LCACOUNT	3.00	2.00	50.00	3.59	1.77	1.00	2.08	3.18	4.69	8.33
LCACOUNT	3.00	3.00	50.00	3.66	1.83	1.00	2.12	3.11	4.86	7.89
LCACOUNT	3.00	4.00	50.00	3.67	1.74	1.00	2.22	3.24	5.16	7.64
LCACOUNT	3.00	5.00	50.00	3.60	1.73	1.00	2.14	3.42	4.82	8.32
LCACOUNT	4.00	1.00	50.00	3.62	1.79	1.00	2.12	3.11	4.98	7.29
LCACOUNT	4.00	2.00	50.00	3.55	1.76	1.00	2.06	3.19	4.73	8.09
LCACOUNT	4.00	3.00	50.00	3.58	1.75	1.00	2.01	3.09	4.85	7.73
LCACOUNT	4.00	4.00	50.00	3.63	1.79	1.00	2.03	3.15	5.08	7.36
LCACOUNT	4.00	5.00	50.00	3.58	1.78	1.00	2.03	3.17	4.90	7.34
LCACOUNT	5.00	1.00	50.00	3.59	1.72	1.00	2.12	3.14	4.84	7.90
LCACOUNT	5.00	2.00	50.00	3.60	1.82	1.00	2.14	3.25	4.55	9.07
LCACOUNT	5.00	3.00	50.00	3.51	1.70	1.00	2.11	2.98	4.71	7.94
LCACOUNT	5.00	4.00	50.00	3.59	1.79	1.00	2.00	3.40	4.81	7.69
LCACOUNT	5.00	5.00	50.00	3.59	1.76	1.00	2.13	3.27	4.87	7.76
SPAMX	1.00	1.00	50.00	3.49	1.75	1.00	2.04	3.05	4.98	7.60
SPAMX	1.00	2.00	50.00	3.50	1.79	1.00	2.03	2.88	4.75	7.94
SPAMX	1.00	3.00	50.00	3.53	1.87	1.00	2.06	3.08	4.74	8.75
SPAMX	1.00	4.00	50.00	3.44	1.73	1.04	2.08	2.94	4.64	7.73
SPAMX	1.00	5.00	50.00	3.57	1.79	1.00	2.24	2.94	4.91	7.77
SPAMX	2.00	1.00	50.00	3.49	1.77	1.00	2.11	3.10	4.53	8.09
SPAMX	2.00	2.00	50.00	3.54	1.77	1.04	2.11	3.03	4.58	7.51
SPAMX	2.00	3.00	50.00	3.55	1.75	1.00	2.18	3.17	4.60	7.55
SPAMX	2.00	4.00	50.00	3.53	1.85	1.00	2.09	3.03	4.84	8.03
SPAMX	2.00	5.00	50.00	3.49	1.74	1.00	2.13	3.00	4.76	7.75
SPAMX	3.00	1.00	50.00	3.55	1.84	1.00	2.08	3.00	4.49	8.40
SPAMX	3.00	2.00	50.00	3.51	1.78	1.00	2.13	3.00	4.60	8.33
SPAMX	3.00	3.00	50.00	3.57	1.84	1.00	2.12	2.94	4.73	7.89
SPAMX	3.00	4.00	50.00	3.55	1.77	1.00	2.15	3.15	4.94	7.64
SPAMX	3.00	5.00	50.00	3.52	1.73	1.00	2.14	3.21	4.74	8.32
SPAMX	4.00	1.00	50.00	3.52	1.82	1.00	2.05	2.97	4.98	7.29
SPAMX	4.00	2.00	50.00	3.44	1.78	1.00	2.01	2.92	4.53	8.09
SPAMX	4.00	3.00	50.00	3.48	1.77	1.00	2.01	3.04	4.72	7.73
SPAMX	4.00	4.00	50.00	3.53	1.79	1.00	2.11	3.05	4.97	7.36
SPAMX	4.00	5.00	50.00	3.48	1.80	1.00	2.00	3.12	4.88	7.34
SPAMX	5.00	1.00	50.00	3.48	1.74	1.00	2.07	3.03	4.65	7.90
SPAMX	5.00	2.00	50.00	3.52	1.83	1.00	2.14	3.10	4.48	9.07
SPAMX	5.00	3.00	50.00	3.43	1.71	1.00	2.11	2.94	4.52	7.94
SPAMX	5.00	4.00	50.00	3.50	1.79	1.00	2.01	3.17	4.63	7.69
SPAMX	5.00	5.00	50.00	3.51	1.77	1.00	2.17	3.11	4.73	7.76

Comparison with other programs

The subsequent figures show results for varying parameters for each PP program on eight data sets. Shown is the ND (y-axis, blue), and the eND for those programs that output multiple placement locations (y-axis, orange). Each bar indicates the average accuracy over 100 pruning events. A detailed description of parameters is provided in Suppl. Tab. A.2. The read length of the query sequences was always fixed to 150 bp, with the exception of HIV-104 (additional read lengths of 500 bp), NEOTROP-512 (additional read lengths of 300 bp), and TARA-3748 (additional read lengths of 300 bp).

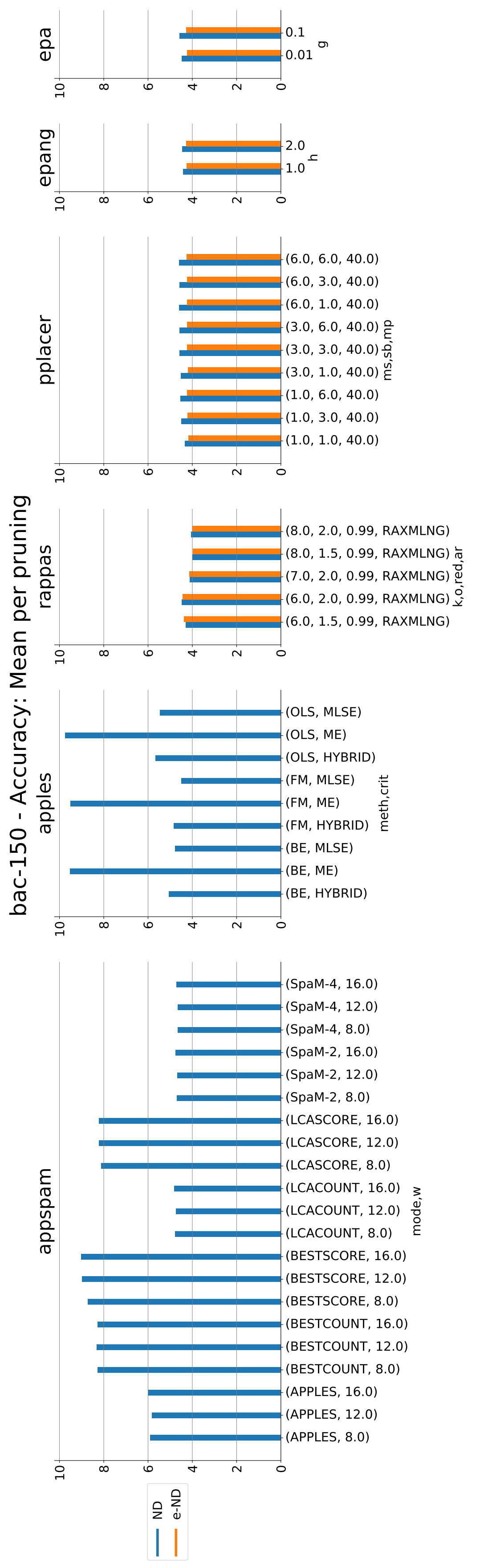


Figure A.2 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the BAC-150 data set. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

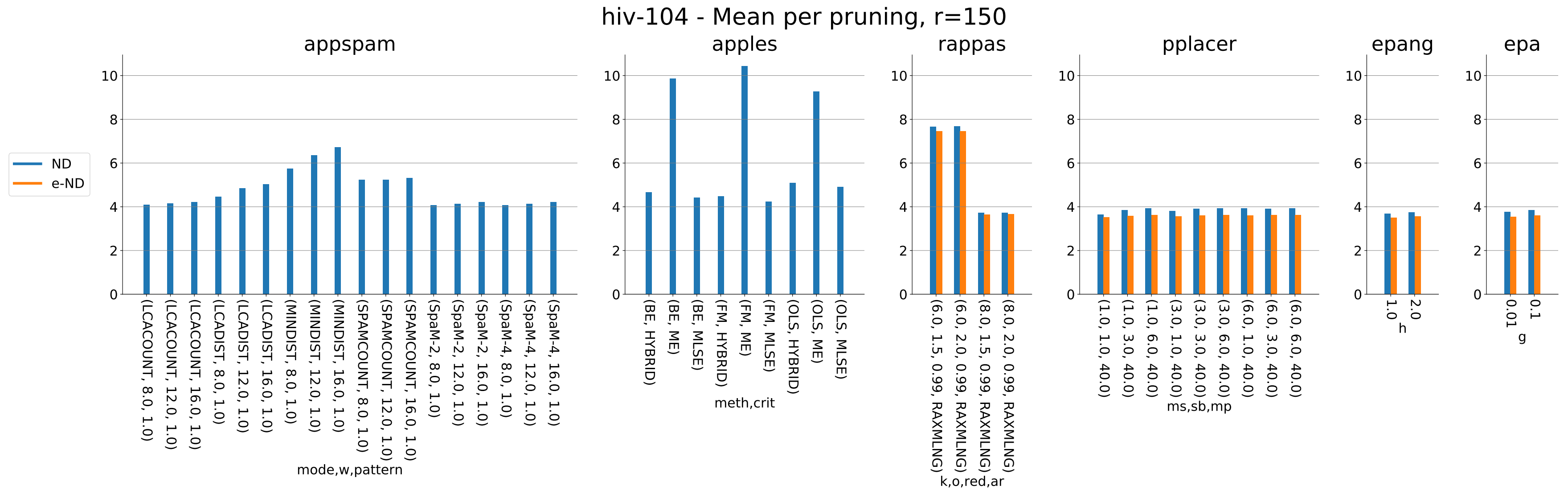


Figure A.3 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the HIV-104 data set. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

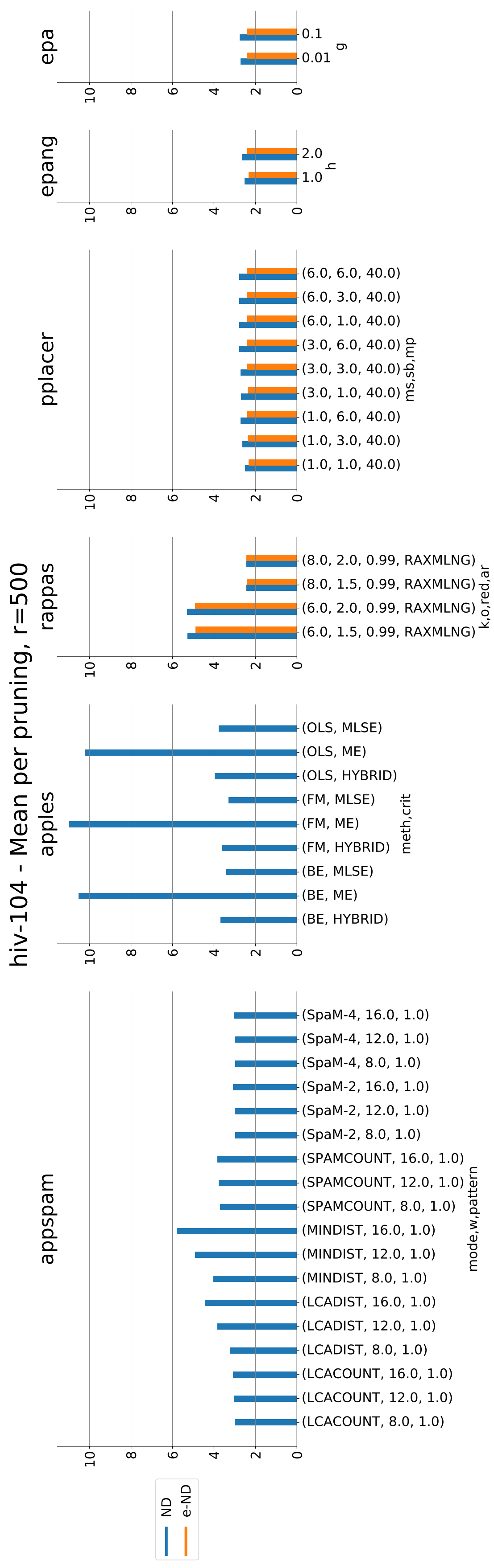


Figure A.4 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the HIV-104 data set. Query read length of 500 bp. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

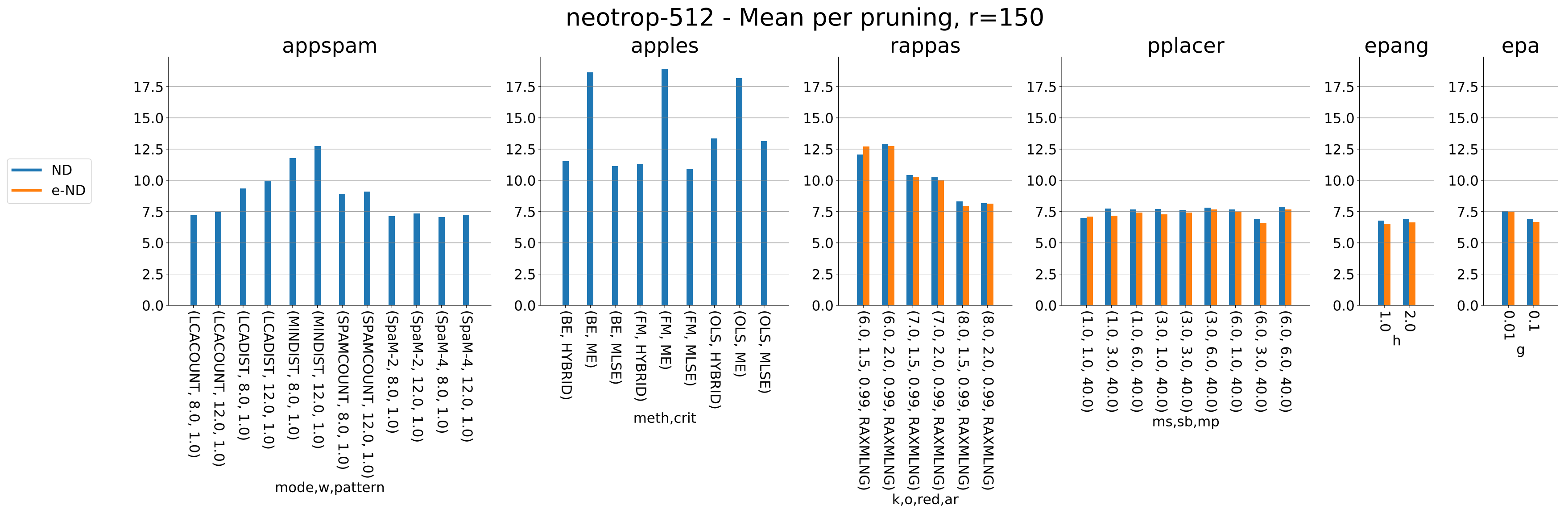


Figure A.5 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the NEOTROP-512 data set. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

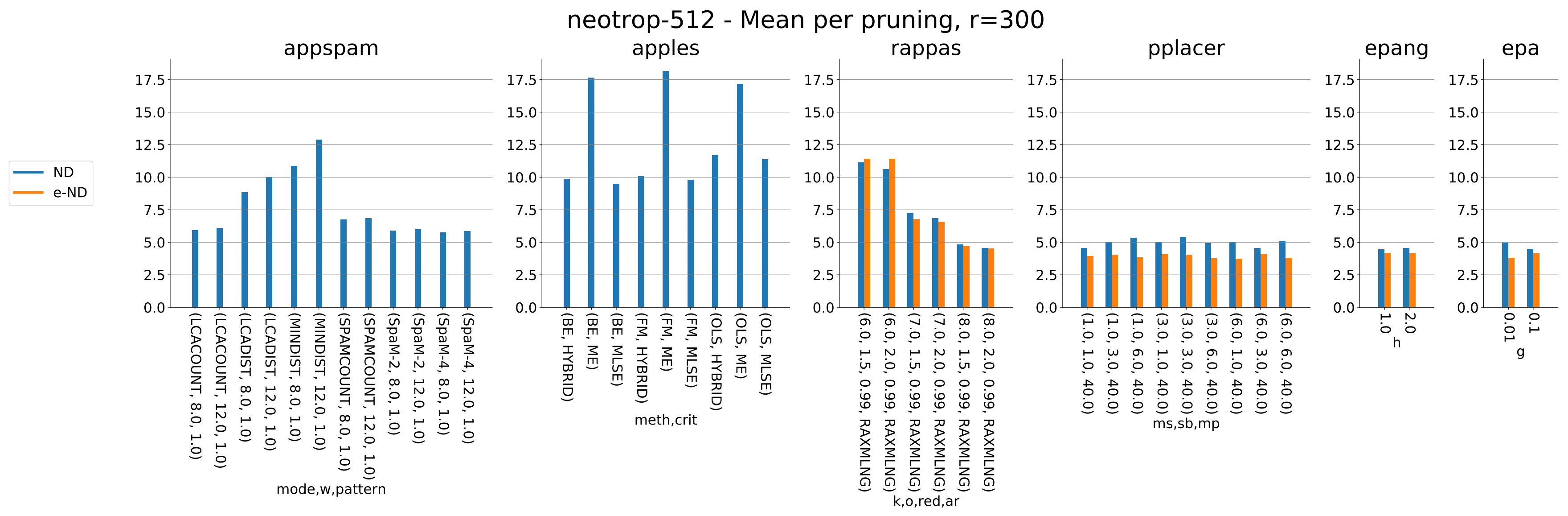


Figure A.6 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the NEOTROP-512 data set. Query read length of 300 bp. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

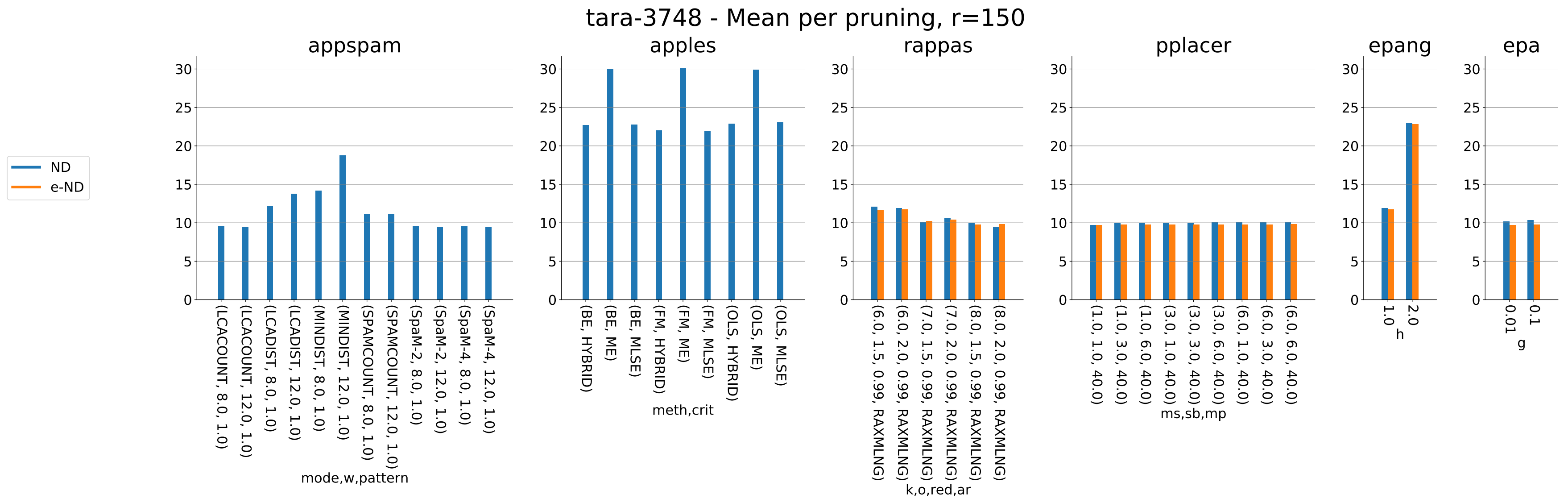


Figure A.7 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the TARA-3748 data set. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

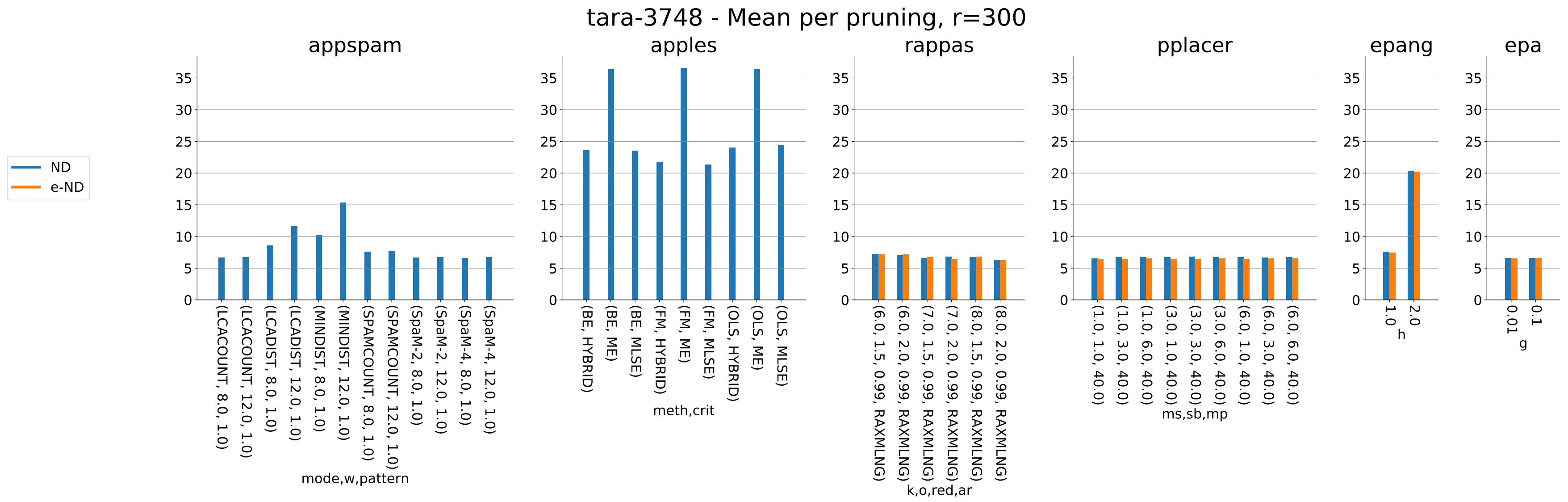


Figure A.8 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the TARA-3748 data set. Query read length of 300 bp. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

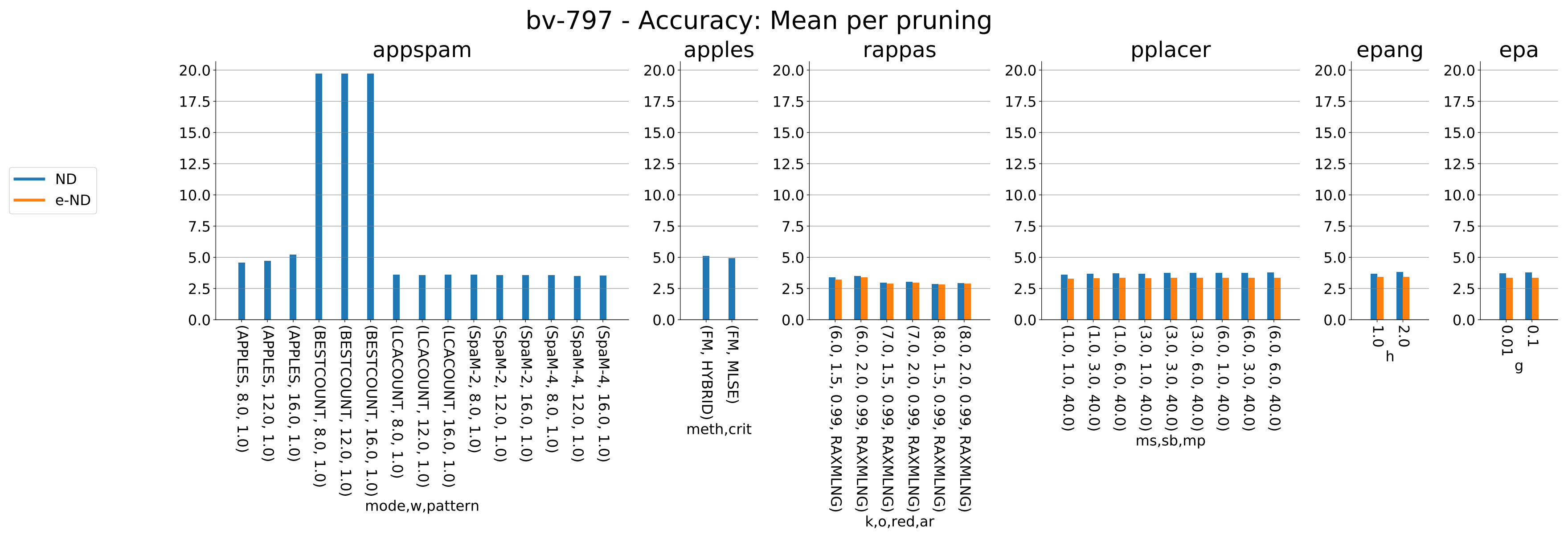


Figure A.9 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the BV-797 data set. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

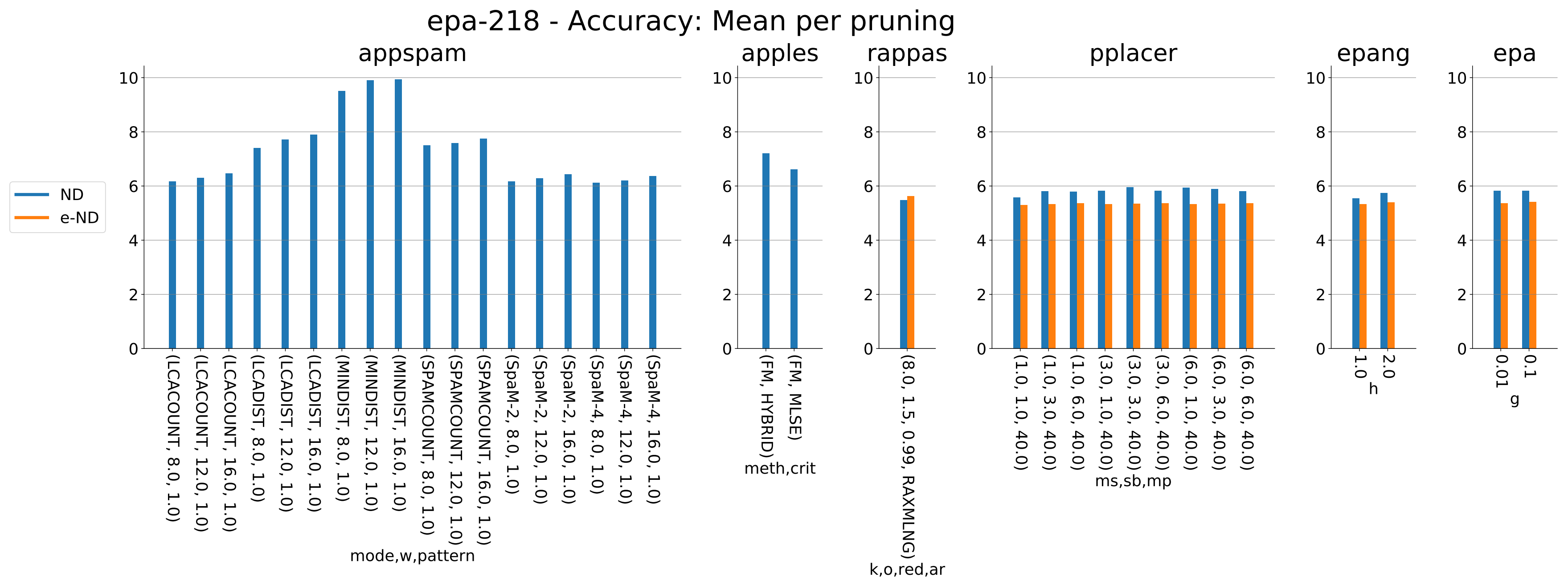


Figure A.10 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the EPA-218 data set. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

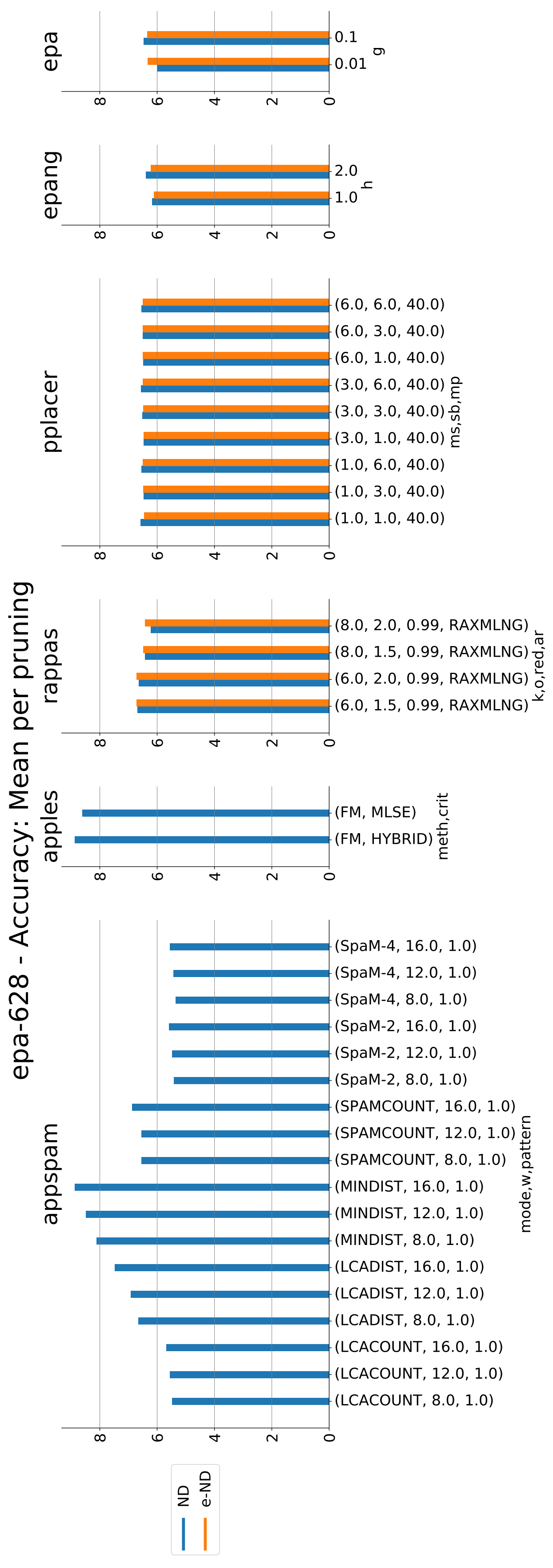


Figure A.11 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the EPA-628 data set. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

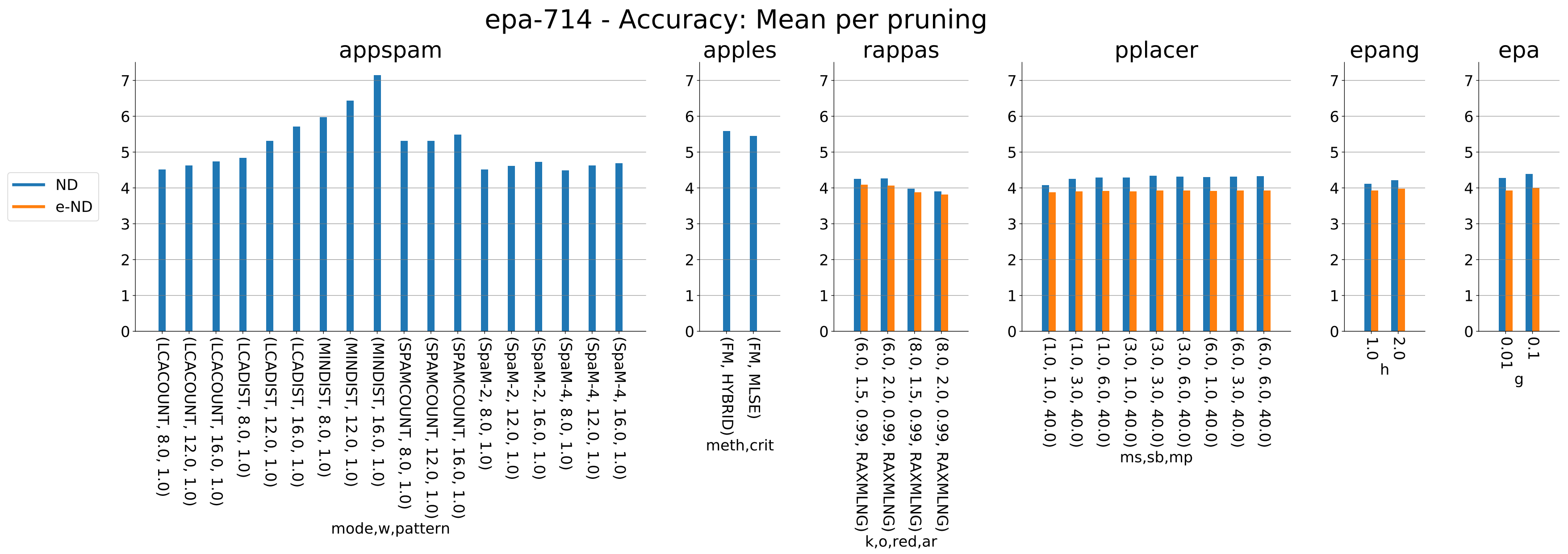


Figure A.12 – Accuracy measured as ND (and eND if applicable) (left y-axes) for multiple PP programs with different parameter combinations (x-axis) on the EPA-714 data set. Adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

Using different read lengths

The following three tables show detailed statistics for the use of different query read lengths for the metataxonomic use case in Sec. 3.1.2. The statistics are divided accordingly to the three plots shown in Fig. 3.10 for three different data sets.

Table A.9 – Summary statistics for the box plots shown in Fig. 3.10. Results are presented for the HIV-104 data set with reads of length 500 bp.

mode	count	mean	std	min	25	50	75	max
appspam-r500	100.00	2.99	0.95	1.01	2.38	2.97	3.42	6.53
pplacer-r500	100.00	2.77	0.88	1.00	2.27	2.57	3.21	5.74
epa-r500	100.00	2.77	0.85	1.00	2.28	2.62	3.18	5.35
epang-r500	100.00	2.52	0.88	1.00	1.90	2.37	3.01	5.77
rappas-r500	100.00	2.43	0.83	1.00	1.81	2.40	2.72	5.32
apples-r500	100.00	3.30	1.21	1.00	2.44	3.10	4.01	7.21

Table A.10 – Summary statistics for the box plots shown in Fig. 3.10. Results are presented for the NEOTROP-512 data set with reads of length 300 bp.

mode	count	mean	std	min	25	50	75	max
appspam-r300	100.00	5.86	3.21	1.00	3.75	5.40	7.17	19.00
pplacer-r300	100.00	4.57	2.78	1.00	2.40	3.62	6.24	14.00
epa-r300	100.00	4.47	2.76	1.00	2.60	3.65	5.40	13.33
epang-r300	100.00	4.45	2.88	1.00	2.40	3.73	5.40	15.00
rappas-r300	10.00	4.82	1.77	2.60	3.65	4.60	5.68	8.46
apples-r300	100.00	9.79	3.99	1.33	6.63	9.48	12.60	22.50

Table A.11 – Summary statistics for the box plots shown in Fig. 3.10. Results are presented for the TARA-3748 data set with reads of length 300 bp.

mode	count	mean	std	min	25	50	75	max
appspam-r300	100.00	6.72	4.97	1.00	3.50	4.75	9.25	30.25
pplacer-r300	100.00	6.69	5.30	1.00	3.25	5.32	9.19	40.00
epa-r300	100.00	6.62	5.53	1.00	3.25	5.50	8.84	45.75
epang-r300	100.00	7.61	6.55	1.00	3.23	5.97	10.41	47.44
rappas-r300	100.00	6.76	6.16	1.00	2.96	4.85	7.82	33.50
apples-r300	100.00	21.30	10.81	4.75	13.44	19.00	28.06	52.25

Using query reads simulated by ART

The following three tables show detailed statistics when query reads are simulated by ART for the metataxonomic use case in Sec. 3.1.2. The statistics are divided accordingly to the three plots shown in Fig. 3.11 for three different data sets.

Table A.12 – Summary statistics for the box plots shown in Fig. 3.11. Query reads were simulated by ART using the Illumina profile on the HIV-104 data set.

mode	count	mean	std	min	25	50	75	max
appspam-r150	50.00	4.06	1.32	2.06	3.11	3.89	4.58	8.62
pplacer-r150	50.00	6.83	1.71	4.33	5.62	6.61	7.55	11.66
epa-r150	50.00	6.60	1.60	4.14	5.60	6.62	7.41	12.14
epang-r150	50.00	6.33	1.62	2.61	5.11	6.23	7.20	11.07
rappas-r150	50.00	9.19	1.78	5.92	7.86	9.27	10.14	14.78
apples-r150	50.00	6.29	2.39	2.14	4.77	6.51	7.48	12.44

Table A.13 – Summary statistics for the box plots shown in Fig. 3.11. Query reads were simulated by ART using the Illumina profile on the NEOTROP-512 data set.

mode	count	mean	std	min	25	50	75	max
appspam-r150	50.00	7.26	2.82	2.38	5.63	6.81	8.56	16.38
pplacer-r150	50.00	12.38	2.03	7.89	11.14	12.23	13.24	17.52
epa-r150	50.00	12.46	2.39	7.86	10.91	12.16	13.61	21.10
epang-r150	50.00	11.56	2.04	6.97	10.49	11.33	12.44	18.27
rappas-r150	50.00	14.50	2.33	9.43	13.22	14.47	15.49	21.55
apples-r150	50.00	15.03	2.73	10.41	12.86	14.48	16.12	21.66

Table A.14 – Summary statistics for the box plots shown in Fig. 3.11. Query reads were simulated by ART using the Illumina profile on the TARA-3748 data set.

mode	count	mean	std	min	25	50	75	max
appspam-r150	50.00	9.61	7.41	2.28	5.09	7.08	10.86	38.90
pplacer-r150	50.00	28.44	10.84	15.30	21.19	25.56	32.36	62.88
epa-r150	50.00	27.61	9.92	16.12	21.35	24.73	30.16	58.42
epang-r150	50.00	25.30	9.96	13.47	18.50	22.71	30.52	58.18
rappas-r150	50.00	24.55	11.38	11.75	16.50	21.17	29.63	57.92
apples-r150	50.00	24.63	11.90	5.80	16.43	21.95	27.53	57.60

A.3 Pruning Difficulty

We proposed two proxies for the *difficulty* of pruning events when performing PEWO's PAC workflow to evaluate PP programs. The first one is the *size* of the pruning event, whereas the size is defined as the total amount of branch lengths that were pruned from $\mathcal{T}_{\text{ref}}$. The second one is the height of the expected placement branch, whereas the height of a node is defined as the number of nodes to the farthest leaf below. Supplementary Figure A.13 and Suppl. Fig. A.14 show scatter plots of these two relationships for all six placement programs on the NEOTROP-512 and BV-797 data sets, respectively. Suppl. Tab. A.15 to Suppl. Tab. A.20 indicate the Spearman correlation coefficient (Spearman CC) and according P values for all respective figures, as indicated.

Table A.15 – Spearman CCs and P values for plots in Fig. 3.15a.

Program	Spearman CC	P Value
App-SpaM	0.25	$1.24 \cdot 10^{-2}$
EPA-ng	0.13	0.20
RAPPAS	0.10	0.30
APPLES	0.17	$9.68 \cdot 10^{-2}$
pplacer	0.19	$5.23 \cdot 10^{-2}$
EPA	0.20	$4.21 \cdot 10^{-2}$

Table A.16 – Spearman CCs and P values for plots in Fig. 3.15b.

Program	Spearman CC	P Value
App-SpaM	-0.18	$6.77 \cdot 10^{-2}$
EPA-ng	$-9.07 \cdot 10^{-2}$	0.37
RAPPAS	$-8.07 \cdot 10^{-2}$	0.43
APPLES	$-9.63 \cdot 10^{-2}$	0.34
pplacer	$-7.77 \cdot 10^{-2}$	0.44
EPA	-0.11	0.26

Table A.17 – Spearman CCs and P values for plots in Suppl. Fig. A.13a.

Program	Spearman CC	P Value
App-SpaM	0.53	$9.91 \cdot 10^{-9}$
EPA-ng	0.67	$2.52 \cdot 10^{-14}$
RAPPAS	0.70	$8.68 \cdot 10^{-16}$
APPLES	0.33	$8.29 \cdot 10^{-4}$
pplacer	0.63	$2.42 \cdot 10^{-12}$
EPA	0.67	$1.41 \cdot 10^{-14}$

Table A.18 – Spearman CCs and P values for plots in Suppl. Fig. A.13b.

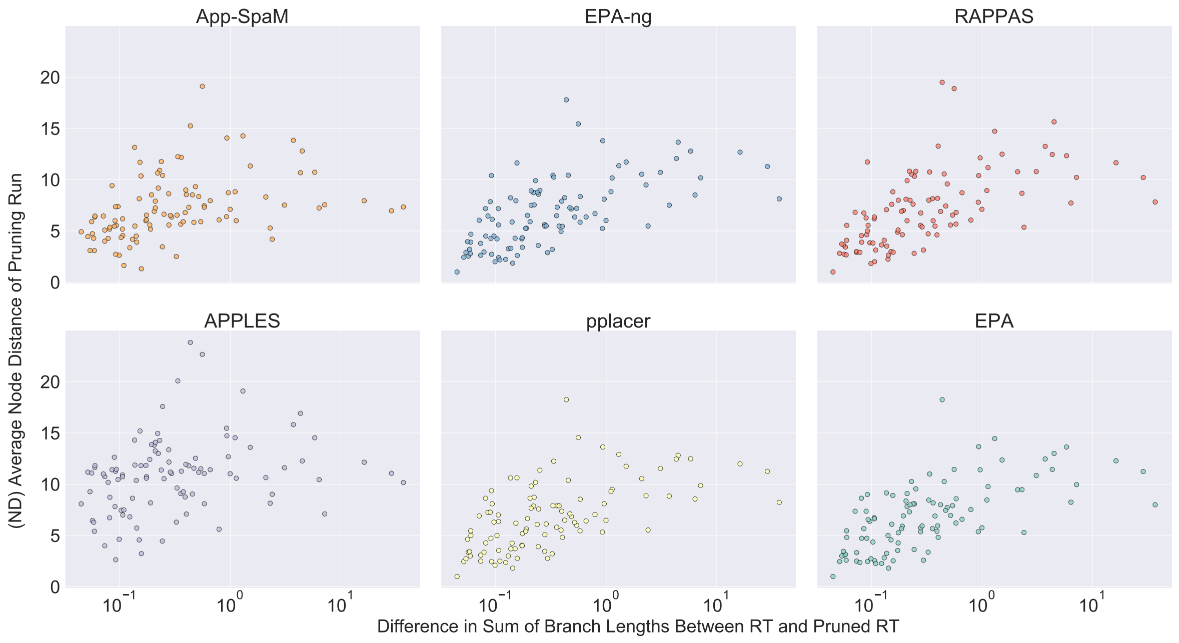
Program	Spearman CC	P Value
App-SpaM	$-4.73 \cdot 10^{-2}$	0.64
EPA-ng	-0.12	0.22
RAPPAS	-0.18	$6.77 \cdot 10^{-2}$
APPLES	$-4.57 \cdot 10^{-2}$	0.65
pplacer	-0.10	0.30
EPA	-0.14	0.16

Table A.19 – Spearman CCs and P values for plots in Suppl. Fig. A.14a.

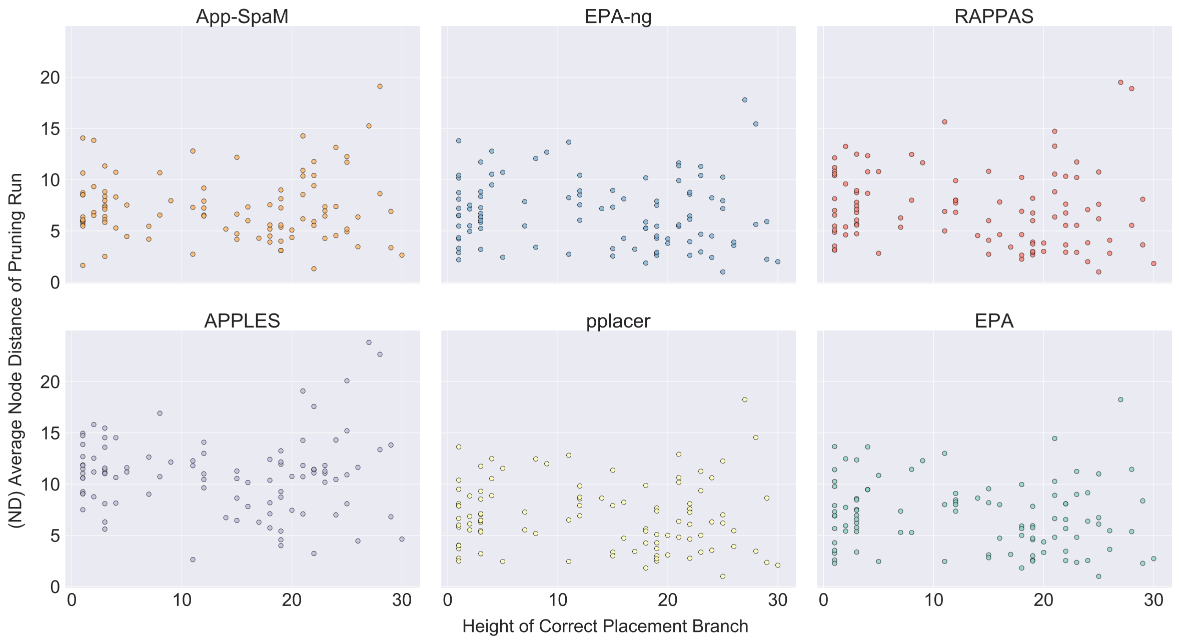
Program	Spearman CC	P Value
App-SpaM	0.28	$5.59 \cdot 10^{-3}$
EPA-ng	0.21	$3.61 \cdot 10^{-2}$
RAPPAS	$2.45 \cdot 10^{-2}$	0.81
APPLES	0.11	0.28
pplacer	0.20	$4.28 \cdot 10^{-2}$
EPA	0.32	$1.09 \cdot 10^{-3}$

Table A.20 – Spearman CCs and P values for plots in Suppl. Fig. A.14b.

Program	Spearman CC	P Value
App-SpaM	0.13	0.20
EPA-ng	0.24	$1.72 \cdot 10^{-2}$
RAPPAS	0.27	$6.04 \cdot 10^{-3}$
APPLES	0.17	$8.29 \cdot 10^{-2}$
pplacer	0.26	$7.88 \cdot 10^{-3}$
EPA	0.16	0.11

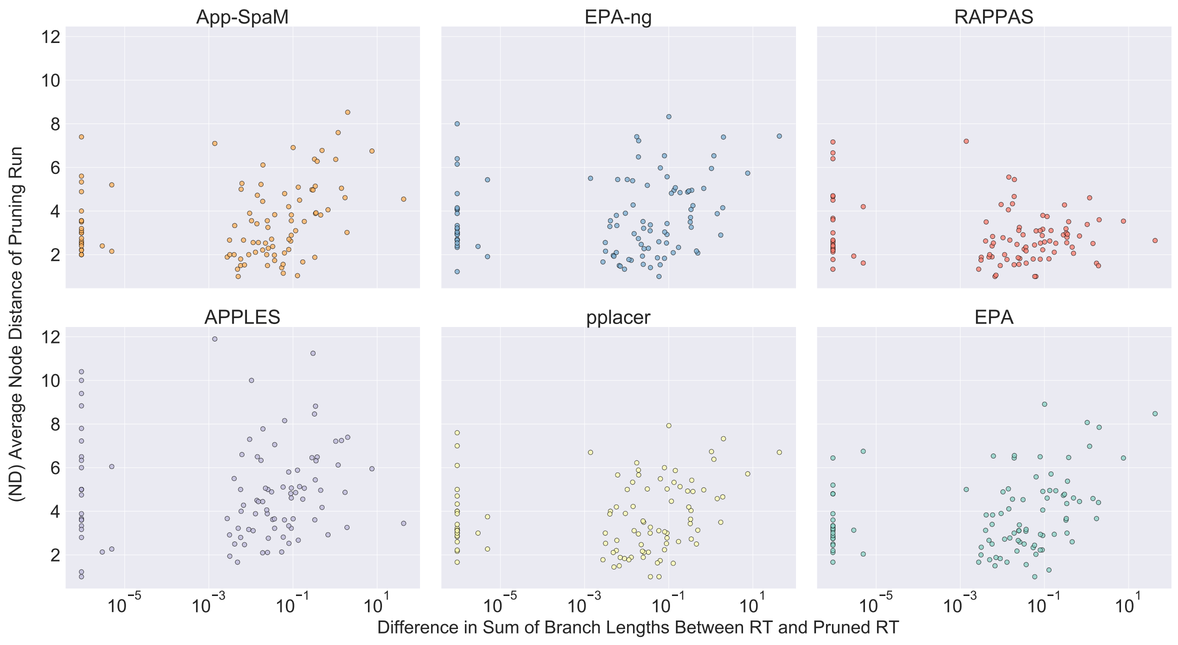


(a) Average node distance (y-axes) of each program (gray boxes) dependent on the difference in the total sum of all branch lengths between $\mathcal{T}_{\text{ref}}$ and the pruned reference tree (x-axes).

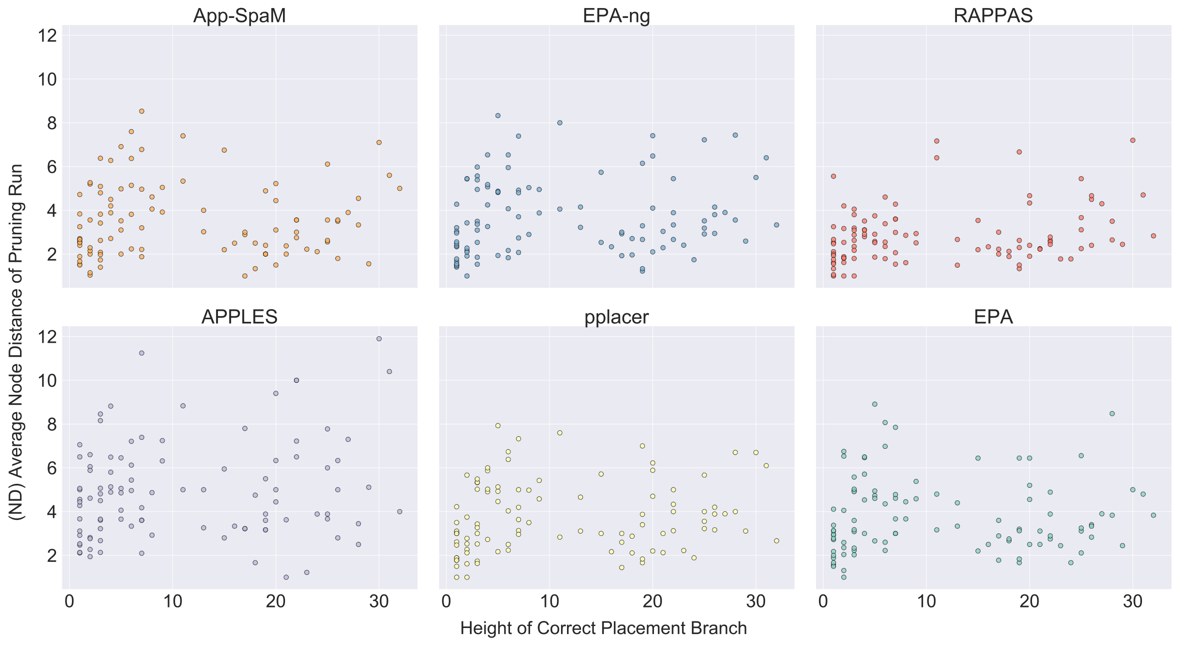


(b) Average node distance (y-axes) of each program (gray boxes) dependent on the level of the expected placement branch (x-axes).

Figure A.13 – Two proxies for the difficulty of each pruning event plotted against the accuracy of placement programs on the NEOTROP-512 data set. This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.



(a) Average node distance (y-axes) of each program (gray boxes) dependent on the difference in the total sum of all branch lengths between $\mathcal{T}_{\text{ref}}$ and the pruned reference tree (x-axes).



(b) Average node distance (y-axes) of each program (gray boxes) dependent on the level of the expected placement branch (x-axes).

Figure A.14 – Two proxies for the difficulty of each pruning event plotted against the accuracy of placement programs on the BV-797 data set. This figure was adapted from the Supplementary Material of *App-SpaM: Phylogenetic placement of short reads without sequence alignment*, in: Bioinformatics Advances, 2021.

Chapter B

Revisiting App-SpaM

This chapter contains supplementary material for additional experiments that we performed for App-SpaM in Chpt. 4. This comprises an analysis of the accuracy of App-SpaM's distance estimates in Suppl. Sec. B.1, spaced-word histograms in Suppl. Sec. B.1, additional information about the sampling of spaced words in Suppl. Sec. B.2, and information on assessing placement uncertainty in Suppl. Sec. B.3.

B.1 Estimating Evolutionary Distances

By default, App-SpaM estimates evolutionary distances by applying the Jukes-Cantor formula to the number of substitutions per sequence position, which is calculated from the don't care positions of all spaced-word matches. Besides Jukes-Cantor, we tested whether using the 2-parameter Kimura model improves distance estimates; the procedure is explained in detail in Sec. 4.1. The subsequent figures show additional examples for the deviation of distance estimates on different data sets. We denote App-SpaM's distance estimate using the Jukes-Cantor model as d_{JC} and, accordingly, App-SpaM's estimate using the Kimura 2 parameter model as d_{K80}. Additionally, we compare the results with 'true' distances estimated from multiple sequence alignments. Hence, $d_{\mathrm{MSA(JC)}}$ and $d_{\mathrm{MSA(K80)}}$ indicate the Jukes-Cantor and Kimura estimates based on alignments, respectively. For each plot, missing dots indicate that no spaced-word match was found between the respective query and reference. Suppl. Fig. B.1 complements the results on the hiv-104 data set shown in Sec. 4.1. Suppl. Fig. B.2 and Suppl. Fig. B.3 show how distance estimates deviate from the ground truth on the bac-150 data set. Suppl. Fig. B.4 to Suppl. Fig. B.6 show results for neotrop-512.

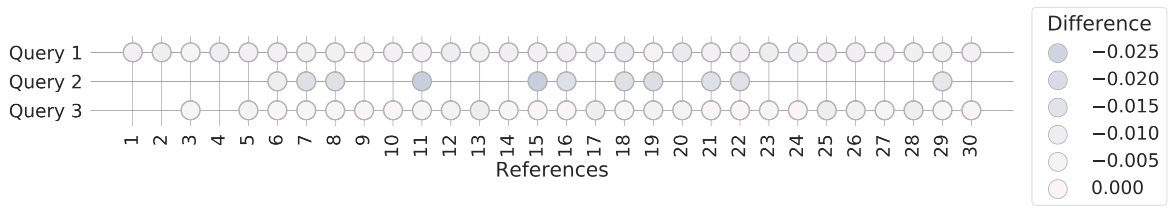


Figure B.1 – Difference in APP-SPAM's distance estimates between using the Jukes-Cantor model and the 2-parameter Kimura model on the HIV-104 data set. Shown is $d_{\mathrm{JC}} - d_{\mathrm{K80}}$ (colored dots) in a heatmap for three randomly selected query sequences (y-axis) and 30 randomly selected reference distances (x-axis).

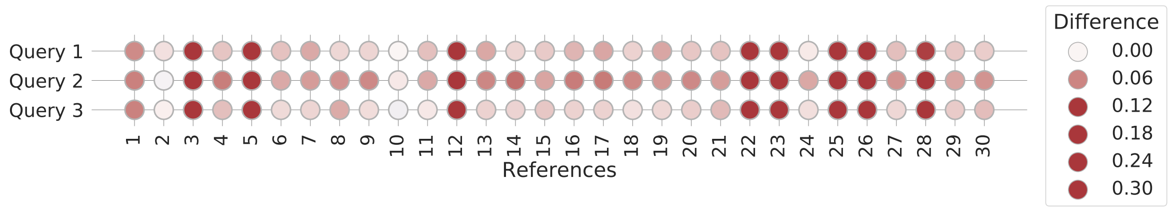


Figure B.2 – Divergence of APP-SPAM's distance estimates under the Jukes-Cantor model from the 'true' distance on the BAC-150 data set. Shown is $d_{\mathrm{MSA(JC)}} - d_{\mathrm{JC}}$ (colored dots) in a heatmap for three randomly selected query sequences (y-axis) and 30 randomly selected reference distances (x-axis).

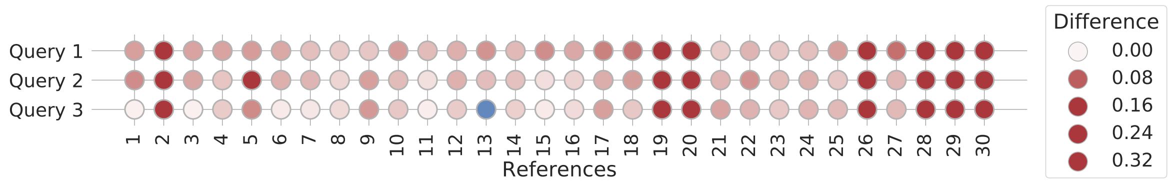


Figure B.3 – Divergence of APP-SPAM's distance estimates under the 2-parameter Kimura model from the 'true' distance on the BAC-150 data set. Shown is $d_{\mathrm{MSA(K80)}} - d_{\mathrm{K80}}$ (colored dots) in a heatmap for three randomly selected query sequences (y-axis) and 30 randomly selected reference distances (x-axis).

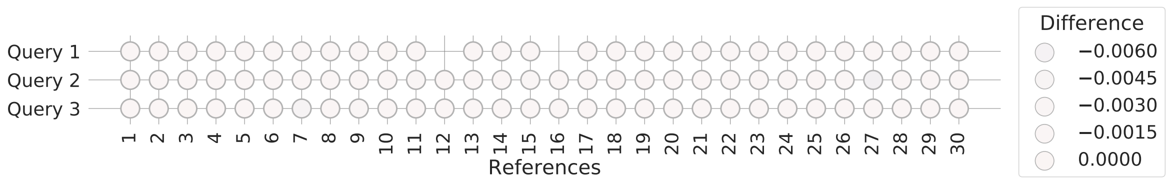


Figure B.4 – Difference in App-SpaM's distance estimates between using the Jukes-Cantor model and the 2-parameter Kimura model on the NEOTROP-512 data set. Shown is $d_{\mathrm{JC}} - d_{\mathrm{K80}}$ (colored dots) in a heatmap for three randomly selected query sequences (y-axis) and 30 randomly selected reference distances (x-axis).

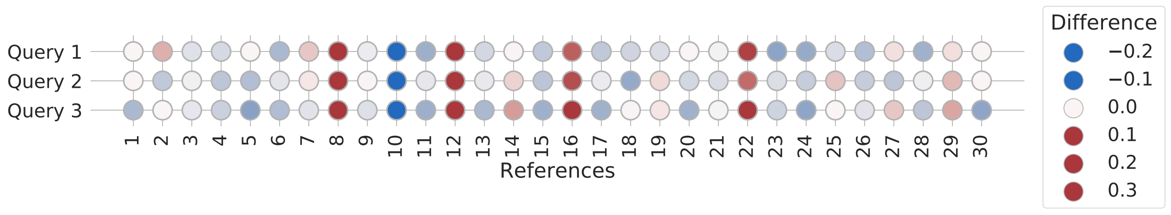


Figure B.5 – Divergence of App-SpaM's distance estimates under the Jukes-Cantor model from the 'true' distance on the NEOTROP-512 data set. Shown is $d_{\mathrm{MSA(JC)}} - d_{\mathrm{JC}}$ (colored dots) in a heatmap for three randomly selected query sequences (y-axis) and 30 randomly selected reference distances (x-axis).

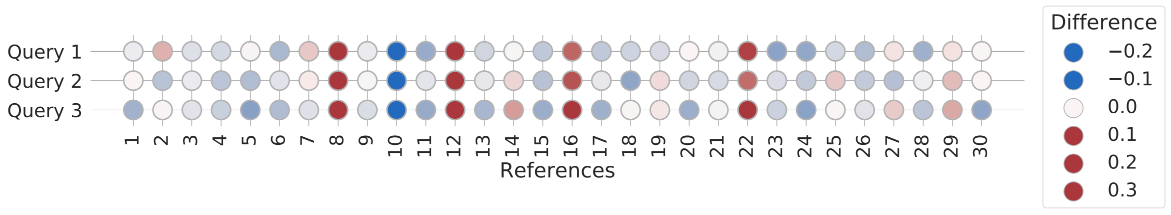


Figure B.6 – Divergence of App-SpaM's distance estimates under the 2-parameter Kimura model from the 'true' distance on the NEOTROP-512 data set. Shown is $d_{\mathrm{MSA(K80)}} - d_{\mathrm{K80}}$ (colored dots) in a heatmap for three randomly selected query sequences (y-axis) and 30 randomly selected reference distances (x-axis).

Spaced-word histograms (spamograms) The score of a spaced-word is defined as the sum over all substitution scores of nucleotides at the don't care positions, given a predefined substitution matrix M. A spaced-word matches histogram (spamogram) displays the score distribution of all spaced-word matches that occurred between the input sequences *before* the filtering procedure is applied. App-SpaM considers *all* spaced-word matches between each query and each reference sequence, thus, for m_r references and m_q queries, this amounts to $m_r \cdot m_q$ sequence pairs. Visualizing a spamogram for all SpaMs independent of their origin is of little value because queries exhibit very different distances to different references. Instead, spamograms are best visualized for all SpaMs between a *single* query sequence and a *single* reference sequence. Spamograms are supposed to show two distinct peaks that we term as the *homologous* peak and *background peak*. The gap between the two peaks as well as the overlap of their respective distributions yields information about the relatedness of the two sequences.

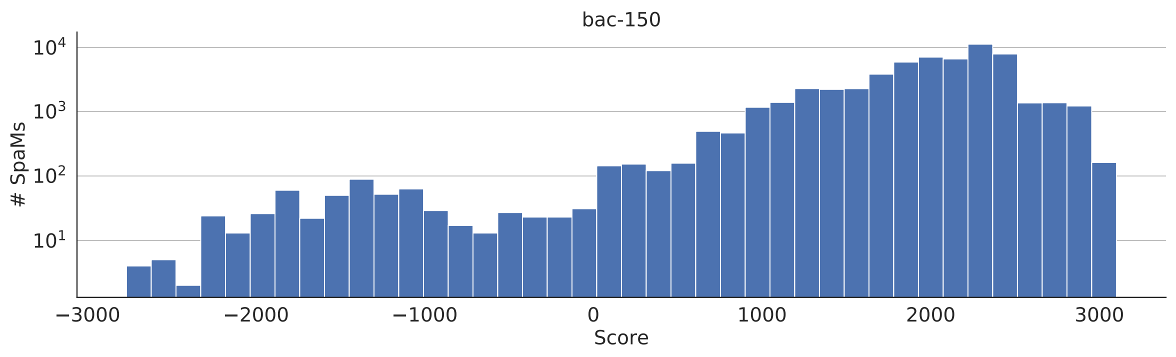


Figure B.7 – Single spamogram for all spaced-word matches between all query and all reference sequences on the bac-150 data set. No clear separation between homologous and background matches is visible.

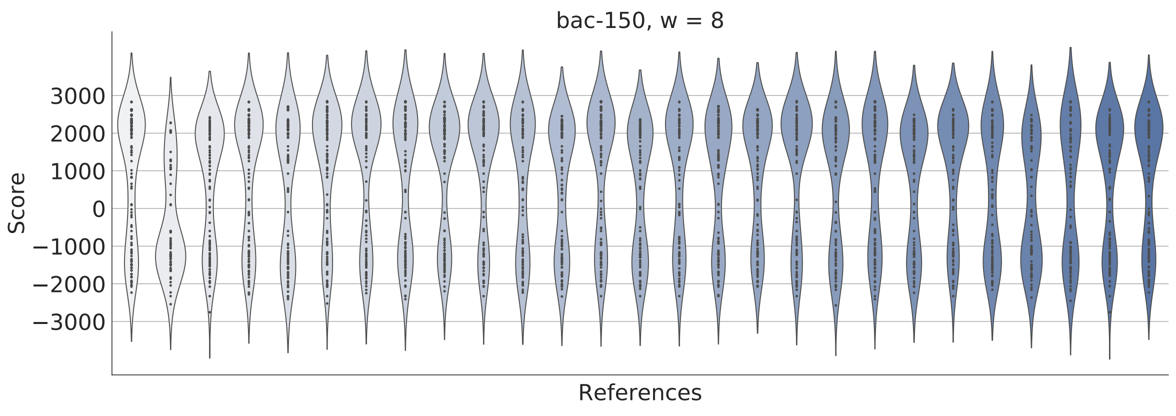


Figure B.8 – Spamograms (each violinplot) for a single query sequence and 20 reference sequences (x-axis) for $w = 8$ on the bac-150 data set.

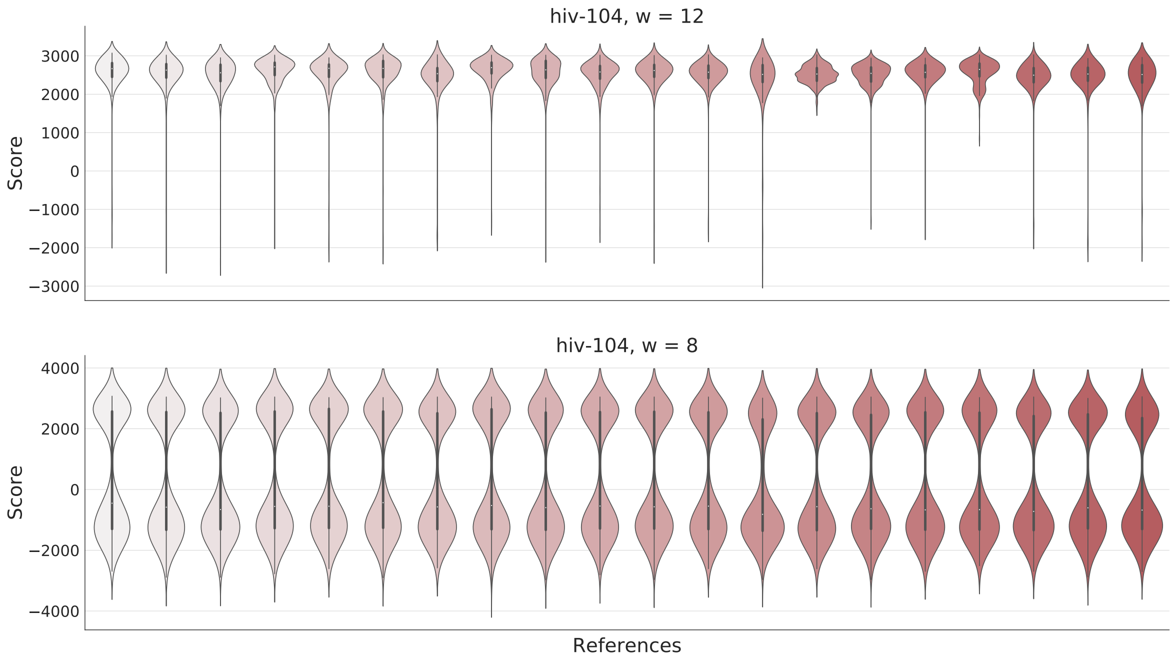


Figure B.9 – Spamograms (each violinplot) for a single query sequence and 20 reference sequences (x-axis) with respect to two different pattern weights on the HIV-104 data set: The top shows spamograms for $w = 12$, the bottom for $w = 8$.

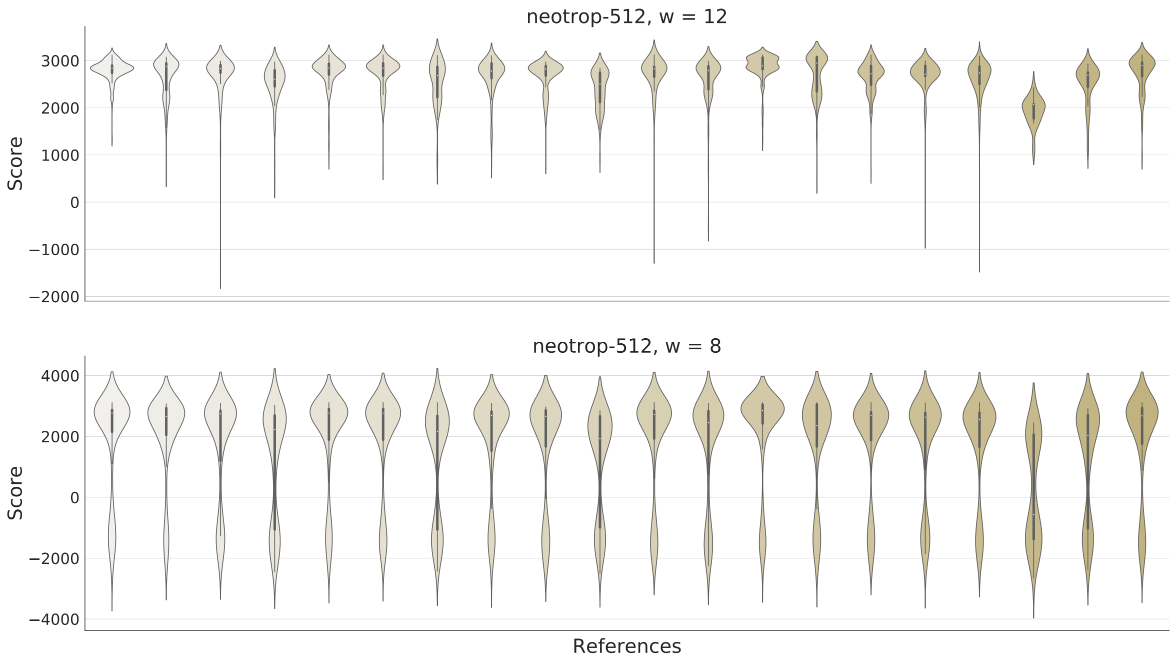


Figure B.10 – Spamograms (each violinplot) for a single query sequence and 20 reference sequences (x-axis) with respect to two different pattern weights on the NEOTROP-512 data set: The top shows spamograms for $w = 12$, the bottom for $w = 8$.

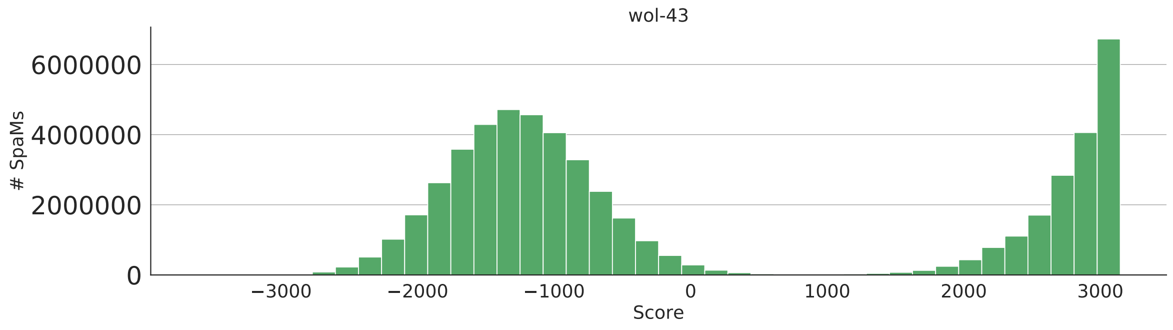


Figure B.11 – Spamogram for SpaMs between *all* queries and *all* references. The background peak is clearly separated from the homologous peak. The background peak extends slightly to the right of score 0, which is the default threshold t for the filtering procedure.

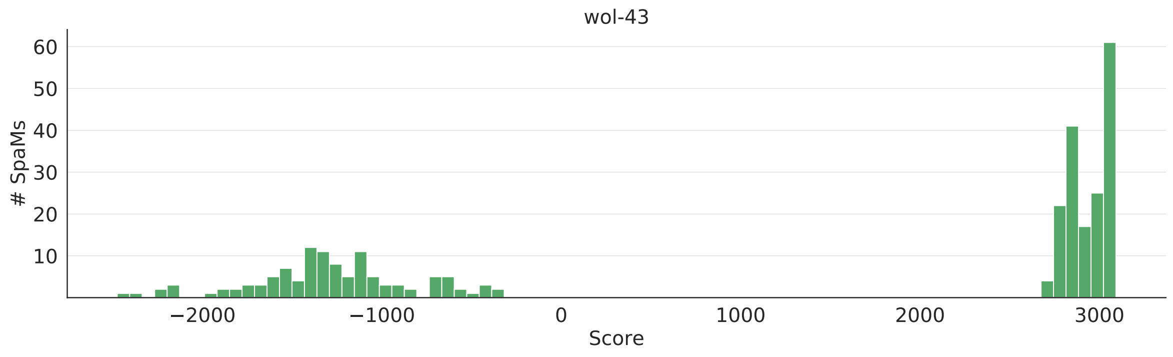


Figure B.12 – Spamogram for SpaMs between a *single* query of length 150 bp and a *single* reference sequence. The background peak is clearly separated from the homologous peak. The total number of spaced-word matches on whose basis the placement is performed is about 160.

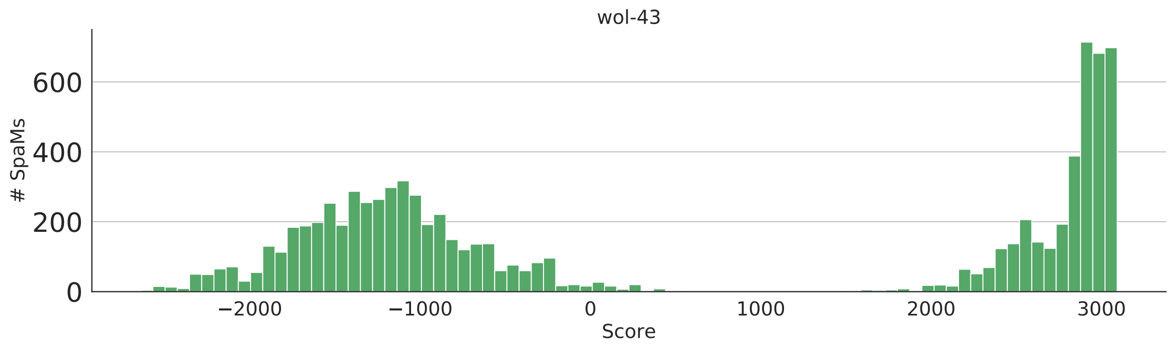


Figure B.13 – Spamogram for SpaMs between a *single* query of length 150 bp and *all* reference sequences. The background peak is clearly separated from the homologous peak. The background peak extends slightly to the right of score 0, which is the default threshold t for the filtering procedure. The homologous spaced-words suggest that multiple peaks exist for the different reference sequences.

B.2 Sampling Spaced Words

The time and memory efficiency of App-SpaM can be improved by performing a hash-based sampling among the spaced-word matches. The sampling procedure ensures that the same spaced words are selected from all input sequences, see Sec. 4.3. In short, the bits representing the match positions of a spaced word (its key K) are scrambled with a random but fixed integer z and a uniform hash function h is applied to $K \oplus z$ whereas $\oplus$ is the bit-wise XOR operator. If the hash value $h(K \oplus z)$ is below a predefined threshold, the spaced word is part of the sample; otherwise, it is discarded. The benefit of sampling within App-SpaM depends on the trade-off between the gain in speed and the loss in accuracy. We discovered that sampling is of limited use for data sets where queries are short, for example, for metagenomic sequencing reads; see Fig. 4.9. The main problem is that the number of remaining spaced words after sampling is not sufficient to specify a sensible placement position. In contrast, sampling is of great use for data sets where query and reference sequences are long; thus, exactly those applications where sampling would be needed. Suppl. Fig. B.14 shows that there is little difference in performance of App-SpaM for varying placement heuristics and pattern weights. Suppl. Fig. B.15 visualizes the loss in accuracy depending on the sampling ratio for two metataxonomic data sets. Lastly, Suppl. Fig. B.15 shows that sampling is of great use if multiple long query reads are present; here, the accuracy stays robust even for very low sampling ratios while the speed is greatly reduced.

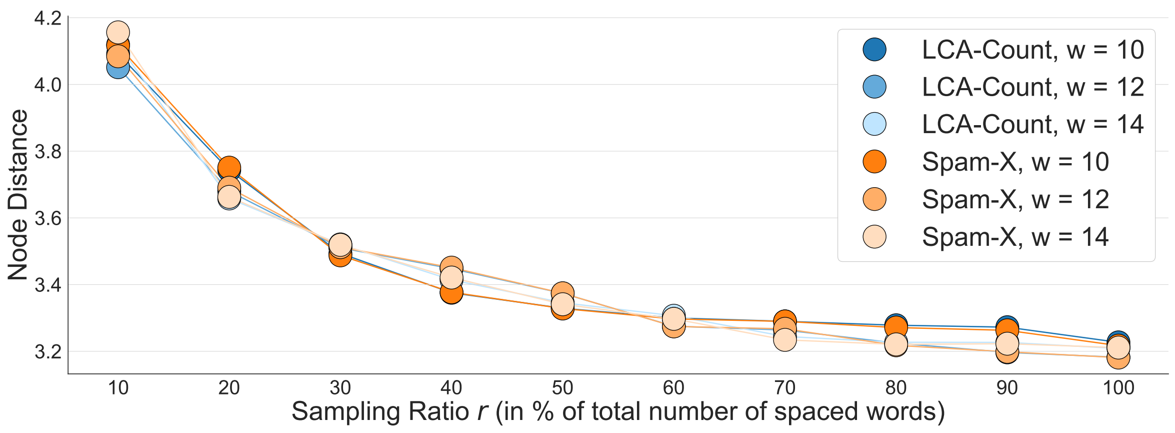


Figure B.14 – Accuracy of App-SpaM depending on the sampling ratio r on the vir-104 data set. The sampling ratio is given as the percentage of spaced words that were used. The average node distance across 50 random prunings (left y-axis) is shown for two different placement modes of App-SpaM with three different weights of the underlying pattern, respectively.

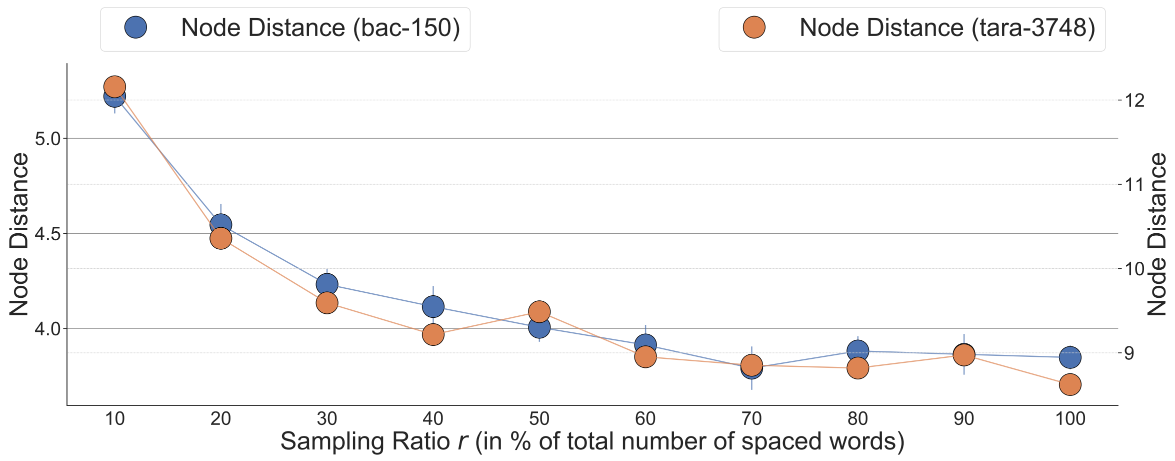


Figure B.15 – Accuracy of App-SpaM depending on the sampling ratio r. The sampling ratio is given as the percentage of spaced words that were used. The average node distance across 50 random prunings is shown for two different data sets: a data set of 150 16S sequences (left y-axis) and a data set of 3 784 16S sequences extracted from ocean water (right y-axis). Standard deviations across the same 6 modes of App-SpaM that were used in Suppl. Fig. B.14 are shown as vertical bars.

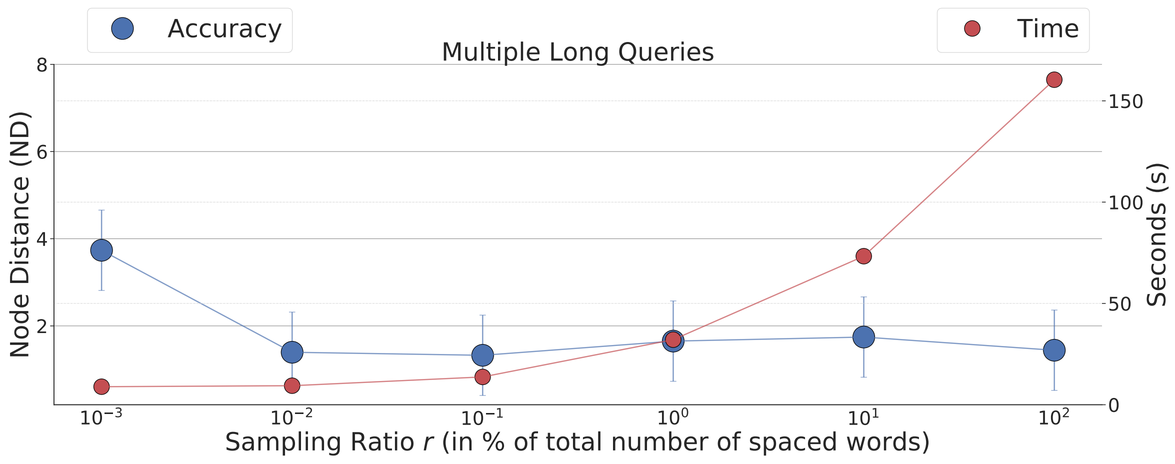


Figure B.16 – Applying sampling to multiple long query sequences on the wol-43 data set. In contrast to other experiments with fewer reads (see Fig. 4.10b), here the gain in speed remains high even when lower sampling ratios are applied.

B.3 Assessing Placement Uncertainty

We discussed the specification of LWR-like placement weights that quantify the uncertainty of placement positions in Sec. 4.4. Here, one problem is the discrepancy that can arise between the number of SpaMs and the distance estimates as visualized in Suppl. Fig. B.17: For some references, only a low number of SpaMs from a highly preserved region are present, which results in low distances, although other sequence regions may only be distantly related.

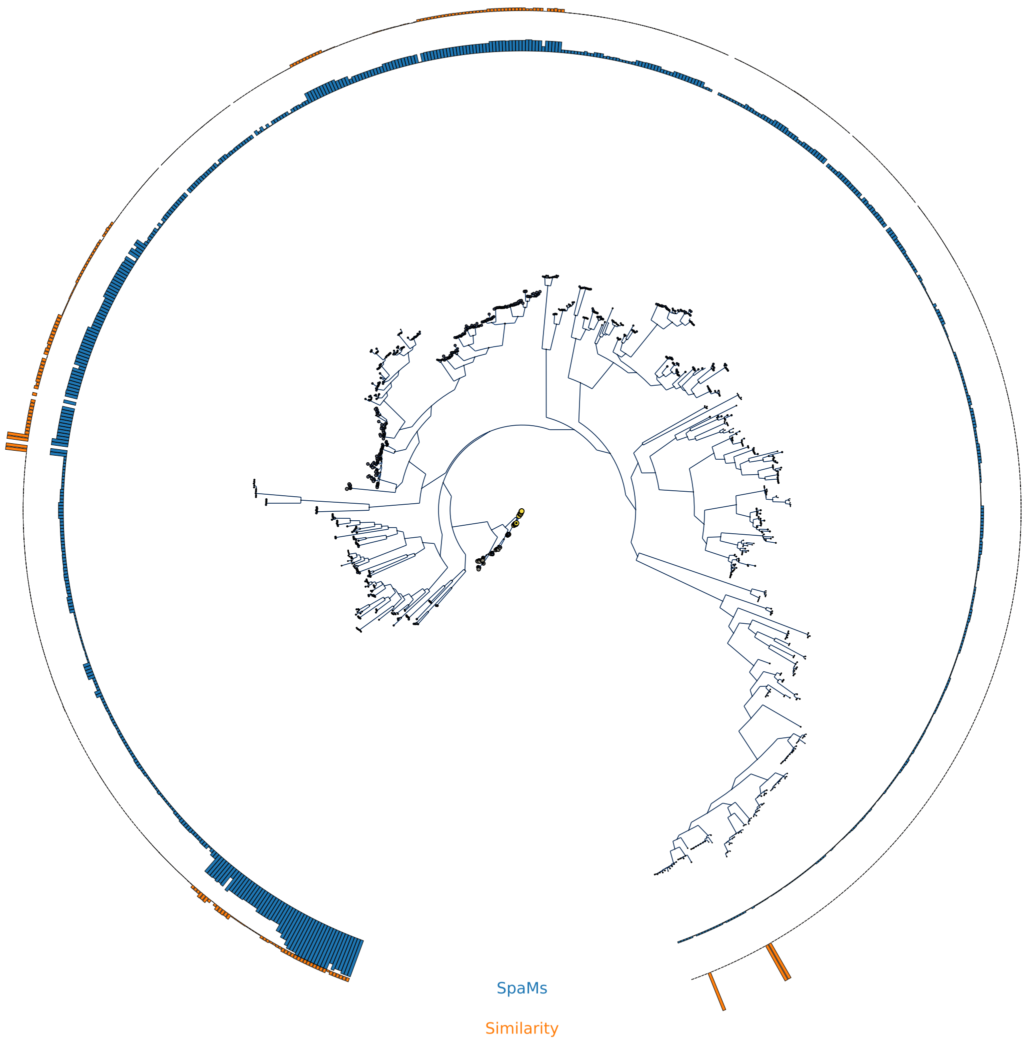


Figure B.17 – Tree with LWR-like values on the BV-797 data set. App-SpaM infers multiple weighted placement positions for a single query S_q on the BV-797 data set. The two outer rings highlight the leaf annotations from App-SpaM's internal algorithm: The bars of the inner ring (blue) indicate the number of spaced word matches between S_q to each reference. The height of each bar is normalized by the maximal SpaMs that occur to any reference sequence. The bars of the outer ring (orange) indicate the the similarity of S_q to each reference.

Chapter C

Continuous Augmentation of Phylogenetic Trees

We carried out additional experiments for the augmentation of phylogenetic trees as described in Sec. 5.1. Supplementary Fig. C.1 shows how results degrade when more HGTs are simulated, Suppl. Fig. C.2 the dependence between pruning size and accuracy. Lastly, Suppl. Fig. C.3 and Suppl. Fig. C.4 show additional results for all four data sets without or with HGTs, respectively.

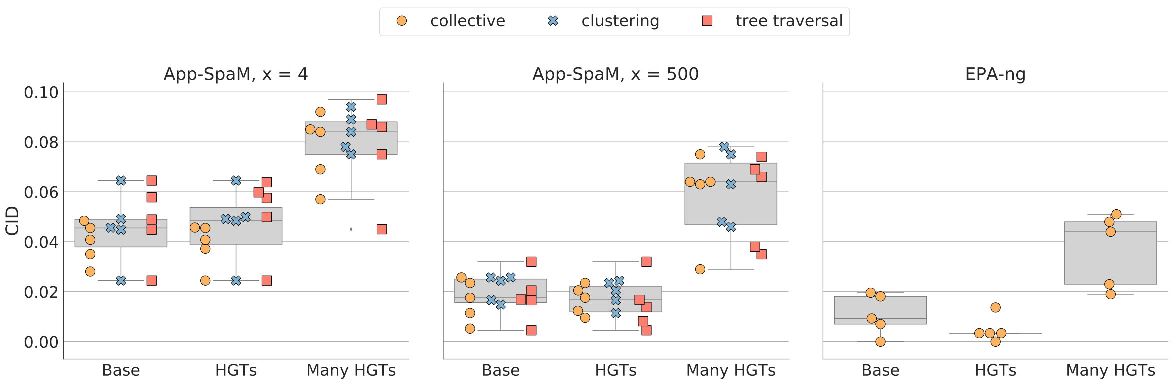


Figure C.1 – Accuracy of augmentation process for App-SpaM and EPA-ng with respect to the number of simulated HGT events. We distinguish between three scenarios: The first has no HGTs (Base), for the second 1% to 2% of all genes have undergone an HGT event (HGTs), and > 4% of all genes are HGTs in the last (Many HGTs).

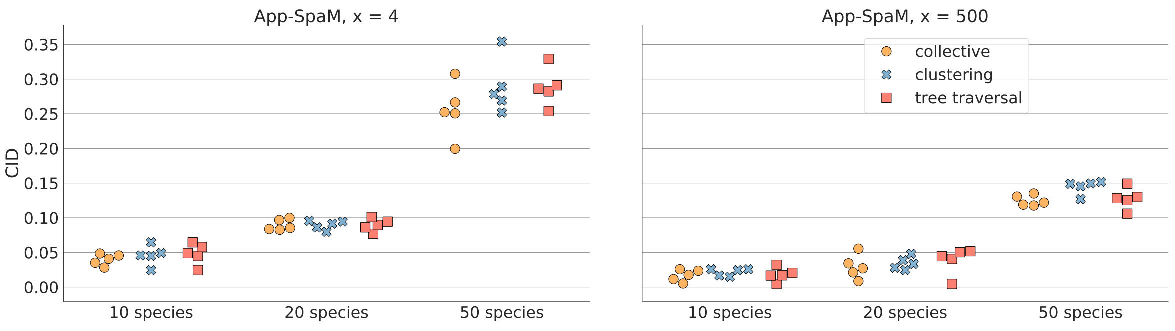


Figure C.2 – Accuracy of augmentation process (CID, y-axis) for App-SpaM with respect to the number of pruned reference sequences (x-axis).

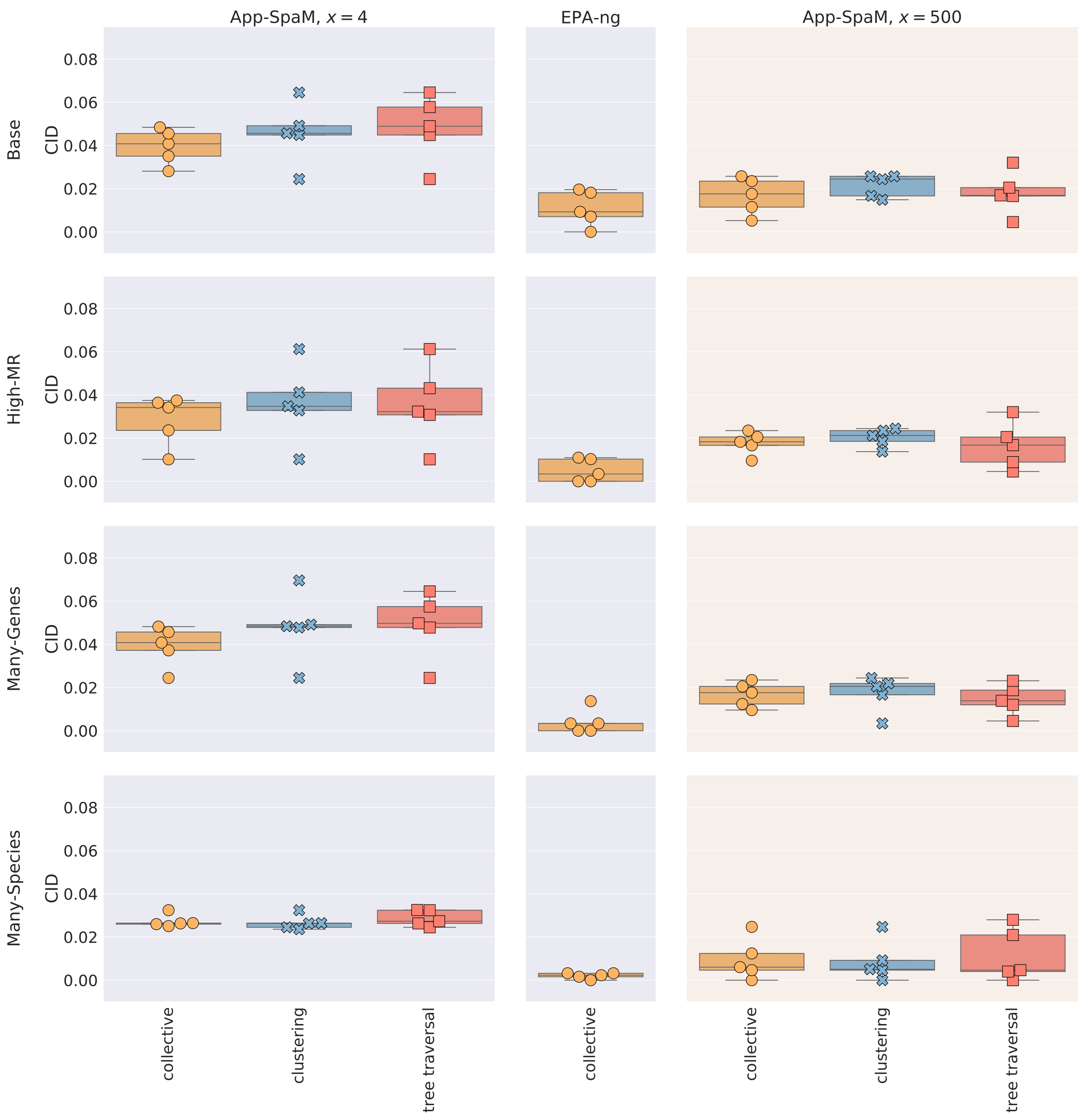


Figure C.3 – Accuracy of augmentation process for App-SpaM and EPA-ng on four artificial data sets without HGT events. For App-SpaM, the three described methods to infer a single species placement position are presented, from left to right: tree traversal (orange), clustering (blue), and collective placement (red). For EPA-ng only the collective placement (red) was performed.

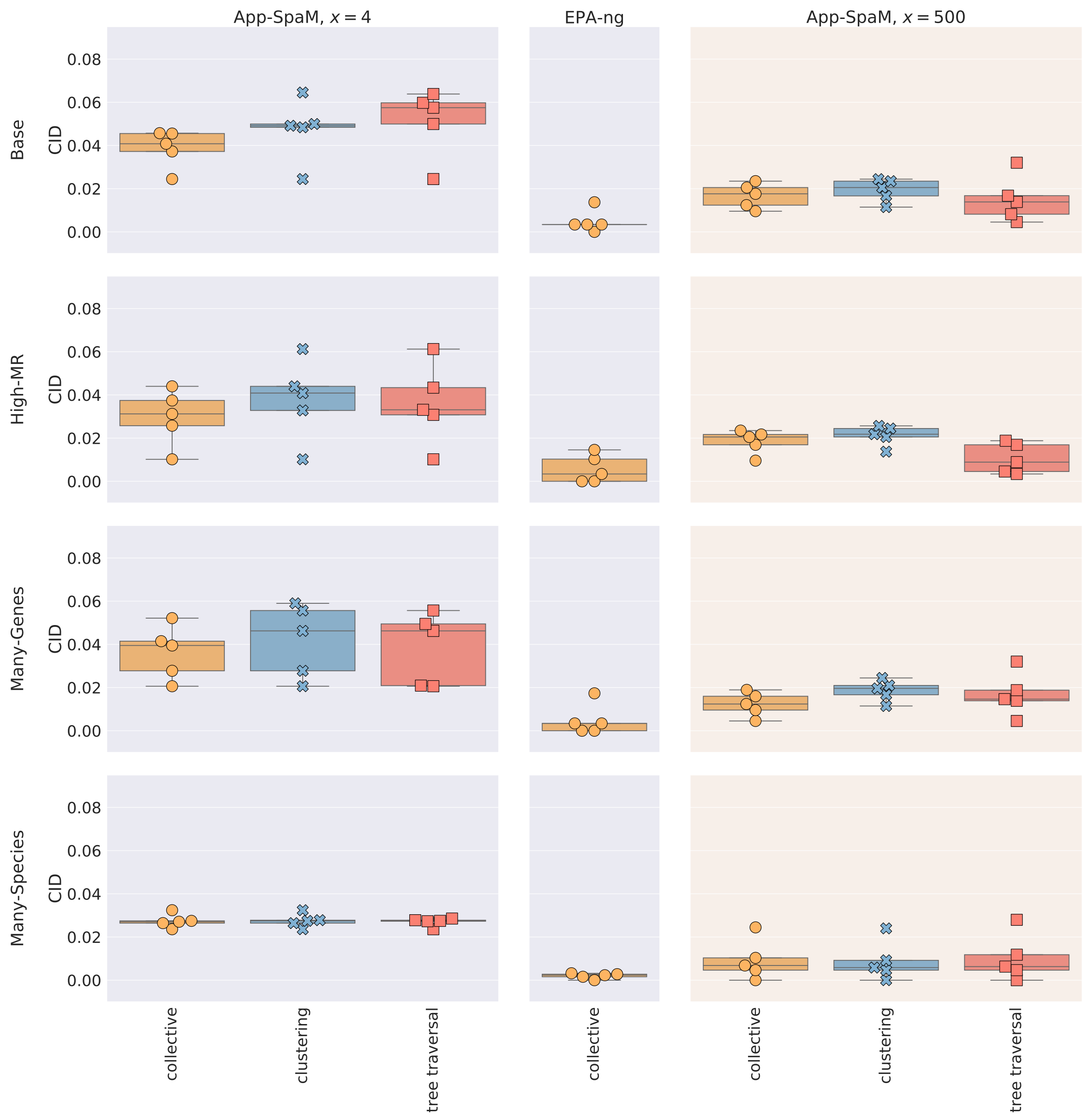


Figure C.4 – Accuracy of augmentation process for APP-SPAM and EPA-NG on four artificial data sets with simulated HGT events. For APP-SPAM, the three described methods to infer a single species placement position are presented, from left to right: tree traversal (orange), clustering (blue), and collective placement (red). For EPA-NG only the collective placement (red) was performed.

C.1 Ordered Augmentation Process

During continuous tree augmentation, new sequences are placed into the original tree in a *random* order. However, the order might have an impact on the quality of resulting trees. For example, placing 'difficult' species might become easier as soon as other related species have already been inserted into the tree. Following this approach, it is necessary to quantify the 'difficulty' of a species placement; we use the uncertainty of the placement as a proxy for its difficulty as discussed earlier. We assess the uncertainty of a species placement from the number, size, and distribution of gene placement clusters across the reference tree. Let C be the set of all placement clusters according to Sec. 4.4. For $c \in C$, let $l(c)$ be the number of branches in c and $w(c)$ the total number of placements in c. Then

$$\frac{\sum_{c \in C} \frac{w(c)}{l(c)}}{|C| \cdot \sum_{c \in C} w(c)} \tag{C.1}$$

describes the overall placement certainty. To perform the ordered augmentation, we first place each species independently on the original tree (without altering it) and calculate the certainty with respect to Eq. C.1. We then sort the queries with respect to their descending certainty and perform the augmentation in this order. The results of the ordered augmentation are shown in Suppl. Fig. C.5. No improvement of the augmented tree is noticeable using this process.

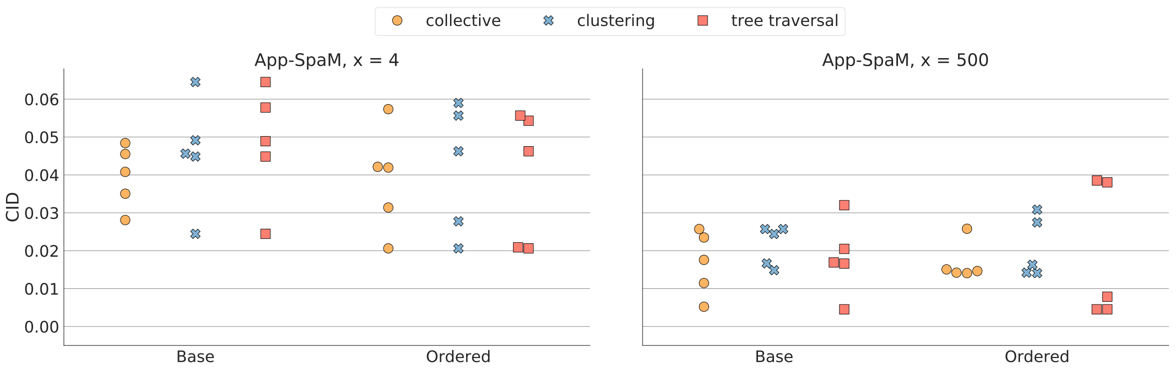


Figure C.5 – Accuracy of augmentation process for App-SpaM with respect to the order in which query organisms are added to the tree. For App-SpaM, the three described methods to infer a single species placement position are presented: tree traversal (orange), clustering (blue), and collective placement (red). In the first case (Base) the sequences are added in a random order as in other experiments. In the second case (Ordered) an order of the sequences is determined from the uncertainty associated with each species based on a preplacement step. No experiments were carried our for EPA-ng due to excessive runtimes.

C.2 Gene and Species Outlier Detection

We detect gene and species outliers using Isolation Forests on feature matrices constructed from the individual gene placement locations during preplacement. For a reference tree with m reference sequences and for m_q species comprising l genes each, the gene feature vectors comprise the following information:

- The number of SpaMs to each reference sequence (m features).
- The estimated phylogenetic distance to each reference sequence (m features).
- The level of its placement within $\mathcal{T}_{\text{ref}}$ (one feature).
- A binary encoding of which reference sequence is contained in the subtree below its placement location (m features).
- Three metrics for the uncertainty of the gene placement, calculated according to Sec. 4.4 (three features).

One such gene feature vector is created for each of the l genes, resulting in a gene feature matrix comprising $3 \cdot m + 4$ features for each species. An Isolation Forest for each such gene feature matrix determines the gene outliers of each species. One overall species feature matrix is created with the following features:

- The variation of gene placement heights in $\mathcal{T}_{\text{ref}}$ (one feature).
- The smallest, largest, and average height among all gene placements in $\mathcal{T}_{\text{ref}}$ (three features).
- The 'compactness' of gene placements in $\mathcal{T}_{\text{ref}}$ (one feature).
- The average, minimal, and maximal number of SpaMs to each reference sequence across all genes ($3 \cdot m$ features).
- The variation of the number of SpaMs to each reference across all genes (m features).
- The average, minimal, and maximal estimated distances to each reference sequence across all genes ($3 \cdot m$).
- The variation of the estimated distances to each reference across all genes (m features).

After running an Isolation Forest on this matrix comprising $8 \cdot m + 5$ features, all outlier species are removed.

Chapter D

A Phylogeny of Old World Monkeys

Plotting spamograms between the Old World monkey species provides many insights into the plausibility of resulting distance estimates. We show two exemplary spamograms between two species in Suppl. Fig. D.1; note, that the x-axis shows the number of mismatches instead of the score in this case to allow for an easy interpretation. The upper spamogram shows a typical distribution with two distinct peaks that represent background and homologous regions. This distribution is only visible when repetitive regions are removed from the species. In contrast, the lower distribution was created without removing repetitive regions. The single visible peak contains spaced-word matches that pass the filtering threshold, but do not originate from homologous regions (but from the repetitive regions instead). The magnitude of this peak is enormous (the non-homologous peak vanishes due to the linear y-scale) because a large number of SpaMs is formed in the repetitive regions.

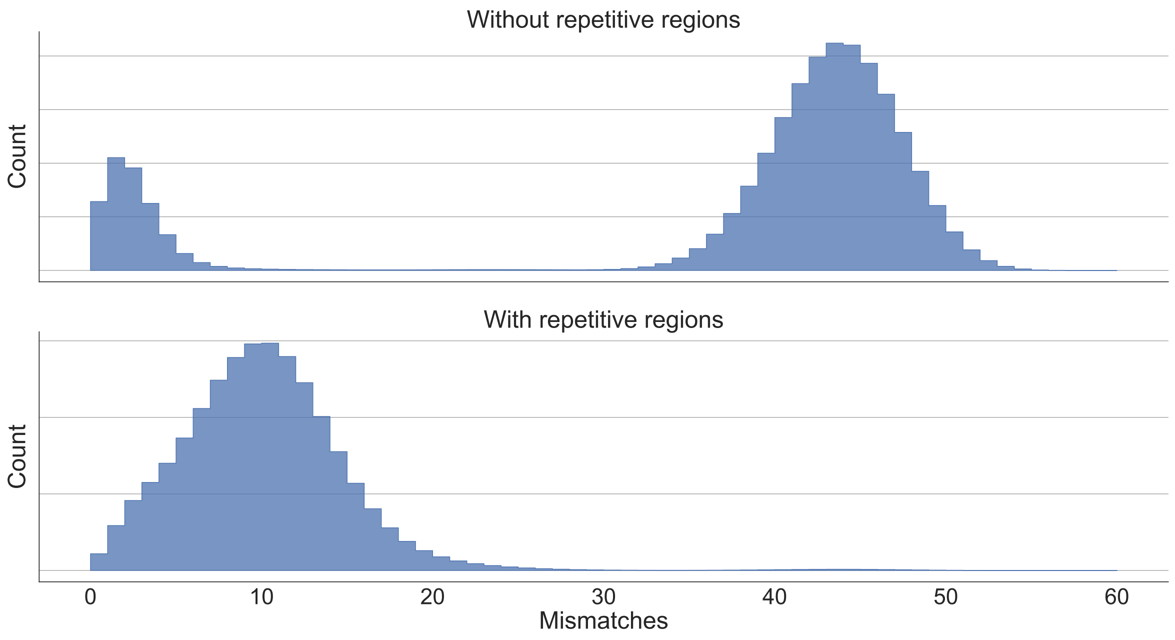


Figure D.1 – Exemplary spamogram between two species without (top) and with (bottom) repetitive regions. In the upper example, all spaced-word matches are used, while a sampling technique was applied in the bottom example. The y-axes indicate the number of SpaMs.

Chapter E

Simon's Congruence

The MaxSimK problem for two sequences S_1 and S_2 is defined as finding the highest k for which the set of all subsequences of length k is identical for S_1 and S_2. In an attempt to utilize MaxSimK for comparing the similarity of biological sequences, we performed several simple experiments that calculate MaxSimK in dependence on the properties of simulated sequences. Suppl. Fig. E.1 shows how MaxSimK behaves between sequences with varying similarity. Suppl. Fig. E.2 to Suppl. Fig. E.4 visualize its dependence on indel and substitution rates. Lastly, Suppl. Fig. E.5 shows how MaxSimK values shift when sequences are encoded as 2-mer sequences instead.

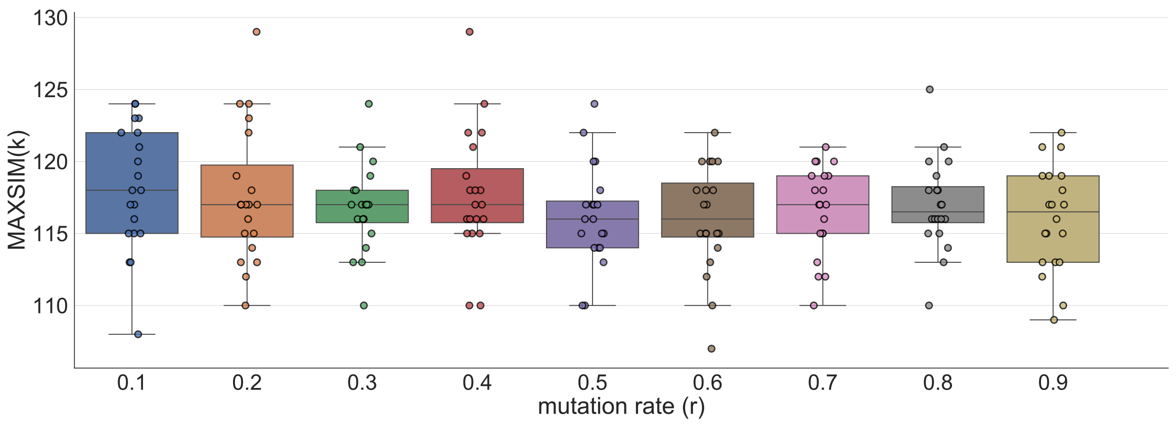


Figure E.1 – Average MaxSimK between sequences with different hamming distances. Mutation rates were were varied for $r \in \{0.1, 0.2, \dots, 0.9\}$.

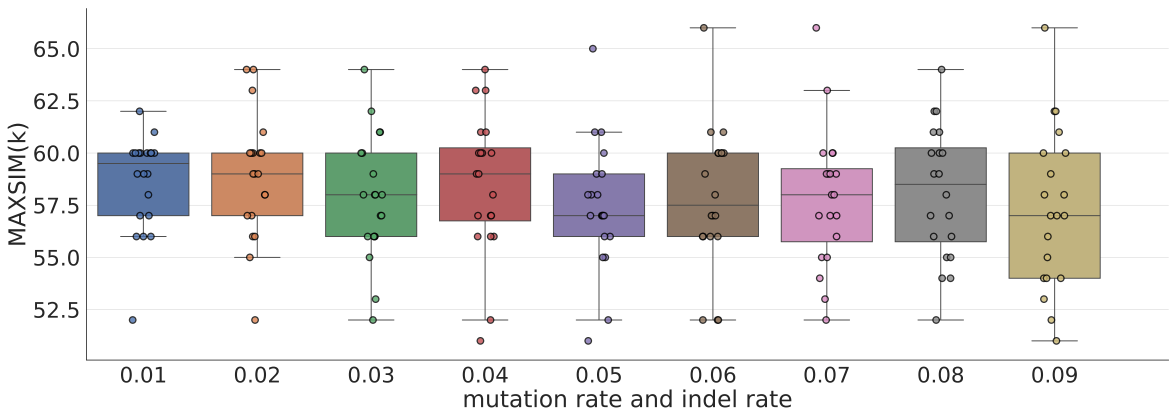


Figure E.2 – MaxSimK between sequences of length 500 with simulated indels. We calculated MaxSimK between sequence pairs for which mutations and indels with a rate r were simulated. Mutation rates were varied from 0.01 up to 0.09 average substitutions per sequence position.

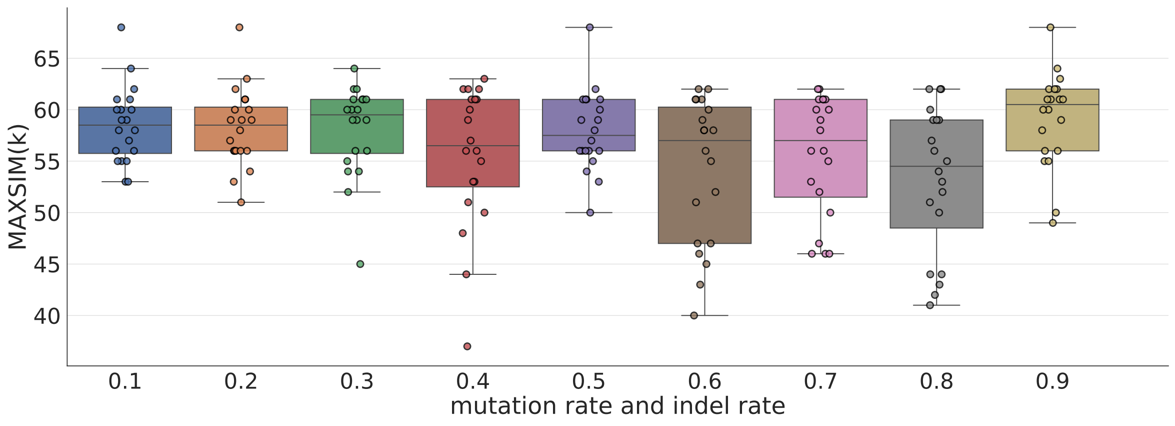


Figure E.3 – MaxSimK between sequences of length 500 with high indel rate. We calculated MaxSimK between sequence pairs for which mutations and indels with a rate r were simulated. Mutation rates were varied from 0.1 up to 0.9 average substitutions per sequence position.

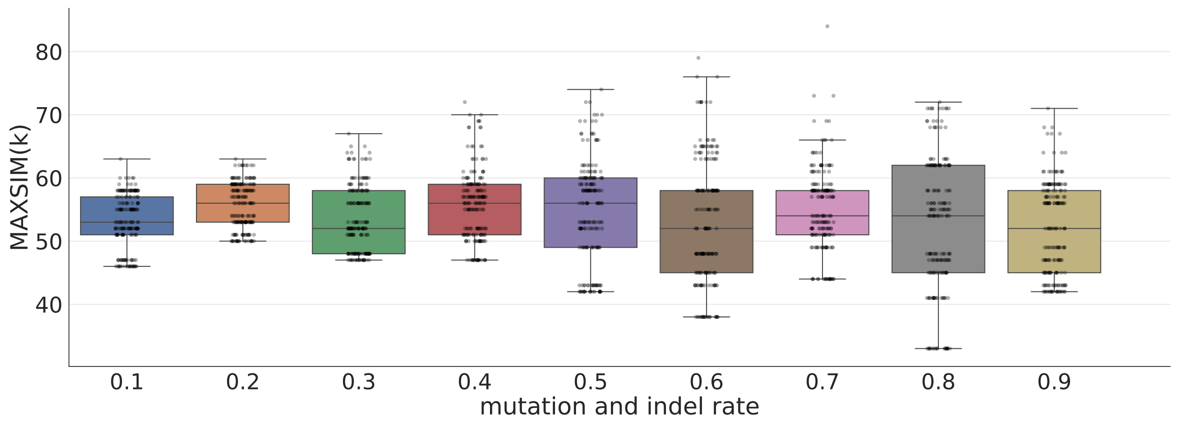


Figure E.4 – MaxSimK between all sequences of length 500 with simulated indels of a higher rate. In contrast to the two preceding figures, here we calculate MaxSimK between each sequence pair for which the same mutation and indel rate r from its respective base sequence applies. Previously, we only calculated the distance between every base sequence and its mutated version. Mutation rates were varied from 0.1 up to 0.9 average substitutions per sequence position.

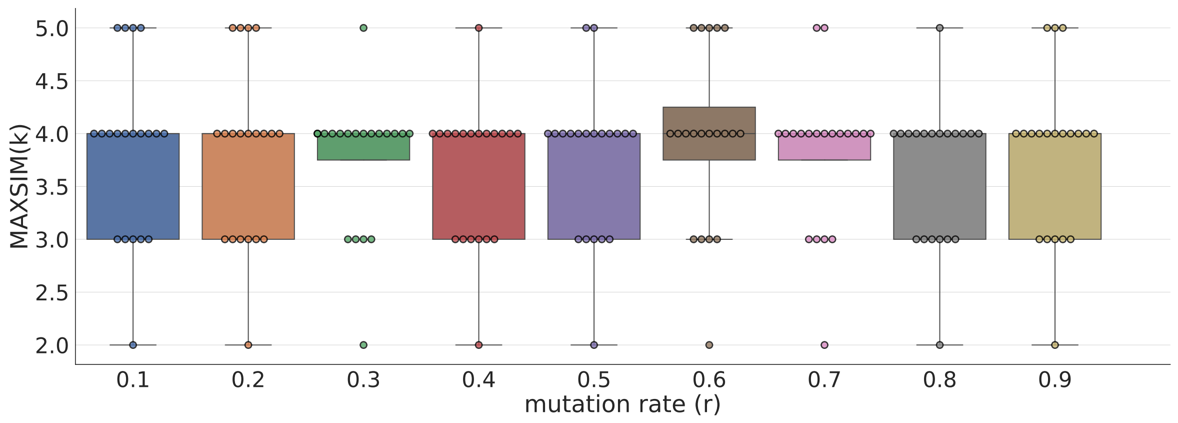


Figure E.5 – Here, the experimental setup is identical to Suppl. Fig. E.1 with a sequence length of $n = 500$. Then, we encoded each sequence by its ordered sequence of 2-mers instead, resulting in sequences of length 250 bp over an alphabet with size $o = 16$. Then we calculated MaxSimK between the adjusted sequences.