Jorge Álvarez Jarreta

Molecular phylogenetic analysis: design and implementation of scalable and reliable algorithms and verification of phylogenetic properties

Departamento

Informática e Ingeniería de Sistemas

Director/es

Mayordomo Cámara, Elvira
Miguel Casado, Gregorio de

ISSN 2254-7606

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

MOLECULAR PHYLOGENETIC ANALYSIS: DESIGN AND IMPLEMENTATION OF SCALABLE AND RELIABLE ALGORITHMS AND VERIFICATION OF PHYLOGENETIC PROPERTIES

Autor

Jorge Álvarez Jarreta

Director/es

Mayordomo Cámara, Elvira

Miguel Casado, Gregorio de

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2017

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

PhD Thesis

Molecular Phylogenetic Analysis: Design and implementation of scalable and reliable algorithms and verification of phylogenetic properties

Author:
Jorge Álvarez Jarreta

Supervisors:
Prof. Elvira Mayordomo Cámara
Dr. Gregorio de Miguel Casado

Universidad Zaragoza

Doctor of Philosophy

March 2017

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Jorge Álvarez Jarreta
March 2017

Resumen

El término *bioinformática* tiene muchas acepciones, una gran parte referentes a la bioinformática molecular: *el conjunto de métodos matemáticos, estadísticos y computacionales que tienen como objetivo dar solución a problemas biológicos, haciendo uso exclusivamente de las secuencias de ADN, ARN y proteínas y su información asociada*. La *filogenética* es el área de la bioinformática encargada del estudio de la relación evolutiva entre organismos de la misma o distintas especies. Al igual que sucedía con la definición anterior, los trabajos realizados a lo largo de esta tesis se centran en la filogenética molecular: la rama de la filogenética que analiza las mutaciones hereditarias en secuencias biológicas (principalmente ADN) para establecer dicha relación evolutiva. El resultado de este análisis se plasma en un árbol evolutivo o *filogenia*. Una filogenia suele representarse como un árbol con raíz, normalmente binario, en el que las hojas simbolizan los organismos existentes actualmente y, la raíz, su ancestro común. Cada nodo interno representa una mutación que ha dado lugar a una división en la clasificación de los descendientes. Las filogenias se construyen mediante procesos de inferencia en base a la información disponible, que pertenece mayoritariamente a organismos existentes hoy en día. La complejidad de este problema se ha visto reflejada en la clasificación de la mayoría de métodos propuestos para su solución como $\mathbb{NP}$-*duros* [9–11].

El caso real de aplicación de esta tesis ha sido el ADN mitocondrial. Este tipo de secuencias biológicas es relevante debido a que tiene un alto

índice de mutación, por lo que incluso filogenias de organismos muy cercanos evolutivamente proporcionan datos significativos para la comunidad biológica. Además, varias mutaciones del ADN mitocondrial humano se han relacionado directamente con enfermedad y patogenias, la mayoría mortales en individuos no natos o de corta edad. En la actualidad hay más de 30000 secuencias disponibles de ADN mitocondrial humano, lo que, además de su utilidad científica, ha permitido el análisis de rendimiento de nuestras contribuciones para datos masivos (*Big Data*). La reciente incorporación de la bioinformática en la categoría Big Data viene respaldada por la mejora de las técnicas de digitalización de secuencias biológicas que sucedió a principios del siglo 21 [15]. Este cambio aumentó drásticamente el número de secuencias disponibles. Por ejemplo, el número de secuencias de ADN mitocondrial humano pasó de duplicarse cada cuatro años, a hacerlo en menos de dos. Por ello, un gran número de métodos y herramientas usados hasta entonces han quedado obsoletos al no ser capaces de procesar eficientemente estos nuevos volúmenes de datos. Este es motivo por el que todas las aportaciones de esta tesis han sido desarrolladas para poder tratar grandes volúmenes de datos.

La contribución principal de esta tesis es un *framework* que permite diseñar y ejecutar automáticamente flujos de trabajo para la inferencia filogenética: *PhyloFlow* [123, 124, 161]. Su creación fue promovida por el hecho de que la mayoría de sistemas de inferencia filogenética existentes tienen un flujo de trabajo fijo y no se pueden modificar ni las herramientas software que los componen ni sus parámetros. Esta decisión puede afectar negativamente a la precisión del resultado si el flujo del sistema o alguno de sus componentes no está adaptado a la información biológica que se va a utilizar como entrada. Por ello, PhyloFlow incorpora un proceso de configuración que permite seleccionar tanto cada uno de los procesos que formarán parte del sistema final, como las herramientas y métodos específicos y sus parámetros. Se han incluido consejos y opciones por defecto durante el proceso de configuración para facilitar su uso, sobre todo a usuarios nóveles. Además, nuestro framework permite la ejecución *desatendida* de los sistemas filogenéticos generados, tanto en ordenadores de sobremesa como en plataformas hardware (*clusters*, *computación en la nube*, ...). Finalmente, se han evaluado las capacidades de PhyloFlow tanto en la reproducción de sistemas de inferencia filogenética publicados anteriormente como en la creación de sistemas orientados a problemas intensivos como el de inferencia del ADN mitocondrial humano. Los resultados muestran que nuestro framework no solo es capaz de realizar los retos planteados, sino que, en el caso de la replicación de sistemas, la posibilidad de configurar cada elemento que los componen mejora ampliamente su aplicabilidad.

Durante la implementación de PhyloFlow descubrimos varias carencias importantes en algunas bibliotecas software actuales que dificultaron la integración y gestión de las herramientas filogenéticas. Por este motivo se decidió crear la primera biblioteca software en Python para estudios de filogenética molecular: *MEvoLib* [28]. Esta biblioteca ha sido diseñada para proveer una sola interfaz para los conjuntos de herramientas software orientados al mismo proceso, como el multialineamiento o la inferencia de filogenias. MEvoLib incluye además configuraciones por defecto y métodos que hacen uso de conocimiento biológico específico para mejorar su precisión, adaptándose a las necesidades de cada tipo de usuario. Como última característica relevante, se ha incorporado un proceso de conversión de formatos para los ficheros de entrada y salida de cada interfaz, de forma que, si la herramienta seleccionada no soporta dicho formato, este es adaptado automáticamente. Esta propiedad facilita el uso e integración de MEvoLib en scripts y herramientas software.

El estudio del caso de aplicación de PhyloFlow al ADN mitocondrial humano ha expuesto los elevados costes tanto computacionales como económicos asociados a la inferencia de grandes filogenias. Por ello, sistemas como PhyloTree [41], que infiere un tipo especial de filogenias de ADN mitocondrial humano, recalculan sus resultados con una frecuencia máxima anual. Sin embargo, como ya hemos comentado anteriormente, las técnicas de secuenciación actuales permiten la incorporación de cientos o incluso miles de secuencias biológicas nuevas cada mes. Este desfase entre productor y consumidor hace que dichas filogenias queden desactualizadas en unos pocos meses. Para solucionar este problema hemos diseñado un nuevo algoritmo que permite la actualización de una filogenia mediante la incorporación iterativa de nuevas secuencias: *PHYSER* [126]. Además, la propia información evolutiva se utiliza para detectar posibles mutaciones introducidas artificialmente por el proceso de secuenciación, inexistentes en la secuencia original. Las pruebas realizadas con ADN mitocondrial han probado su eficacia y eficiencia, con un coste temporal por secuencia inferior a los 20 segundos.

El desarrollo de nuevas herramientas para el análisis de filogenias también ha sido una parte importante de esta tesis. En concreto, se han realizado dos aportaciones principales en este aspecto: *PhyloViewer* [170] y una herramienta para el *análisis de la conservación* [180]. PhyloViewer es un visualizador de filogenias extensivas, es decir, filogenias que poseen al menos un millar de hojas. Esta herramienta aporta una novedosa interfaz en la que se muestra el nodo seleccionado y sus nodos *hijo*, así como toda la información asociada a cada uno de ellos: identificador, secuencia biológica, . . . Esta decisión de diseño ha sido orientada a evitar el habitual "borrón" que se produce en la mayoría de herramientas de visualización al mostrar este tipo de filogenias enteras por pantalla. Además, se ha desarrollado en una arquitectura cliente-servidor, con lo que el procesamiento de la filogenia se realiza una única vez por parte el servidor. Así, se ha conseguido reducir significativamente los tiempos de carga y acceso por parte del cliente. Por otro lado, la aportación principal de nuestra herramienta para el análisis de la conservación se basa en la paralelización de los métodos clásicos aplicados en este campo, alcanzando *speed-ups* cercanos al teórico sin pérdida de precisión. Esto ha sido posible gracias a la implementación de dichos métodos desde cero, incorporando la paralelización a nivel de instrucción, en vez de paralelizar implementaciones existentes. Como resultado, nuestra herramienta genera un informe que contiene las conclusiones del análisis de conservación realizado. El usuario puede introducir un umbral de conservación para que el informe destaque solo aquellas posiciones que no lo cumplan. Además, existen dos tipos de informe con distinto nivel de detalle. Ambos se han diseñado para que sean comprensibles y útiles para los usuarios.

Finalmente, se ha diseñado e implementado un predictor de mutaciones patógenas en ADN mitocondrial desarollado en máquinas de vectores de soporte (SVM): *Mitoclass.1* [189]. Se trata del primer predictor para este tipo de secuencias biológicas. Tanto es así, que ha sido necesario crear el primer repositorio de mutaciones patógenas conocidas, *mdmv.1*, para poder entrenar y evaluar nuestro predictor. Se ha demostrado que Mitoclass.1 mejora la clasificación de las mutaciones frente a los predictores más conocidos y utilizados, todos ellos orientados al estudio de patogenicidad en ADN nuclear. Este éxito radica en la novedosa combinación de propiedades a evaluar por cada mutación en el proceso de clasificación. Además, otro factor a destacar es el uso de SVM frente a otras alternativas, que han sido probadas y descartadas debido a su menor capacidad de predicción para nuestro caso de aplicación.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms / Abbreviations

ABDS	Apache Big Data Stack
BI	Bayesian Inference
BIC	Bayesian information criterion
BLAST	Basic Local Alignment Search Tool
BPMN	Business Process Model and Notation
CBE	Cell Broadband Engine
CDS	Coding DNA sequence
CI	Conservation index
cMI	Cumulative mutual information
DNA	Deoxyribonucleic acid
eUtils	Entrez Programming Utilities
FN	False negative
FP	False positive

FPGA	Field Programmable Gate Array
GPU	Graphics Processor Unit
HGMD	Human Genome Mutation Database
hmtDNA	Human mitochondrial DNA
HPC	High Performance Computing
HTC	High Throughput Computing
HVR	Hypervariable region
iToL	Interactive Tree of Life
MCMC	Markov chain Monte Carlo
mdmv	mtDNA missense variants
MIC	Many Integrated Core
ML	Maximum Likelihood
mmtDNA	Mammal mitochondrial DNA
MP	Maximum Parsimony
MRP	Matrix representation using parsimony
MSA	Multiple sequence alignment
mtDNA	Mitochondrial DNA
NCBI	National Center for Biotechnology Information
nDNA	Nuclear DNA
NGS	Next-generation sequencing

NJ	Neighbor-joining
PHYLIP	Phylogeny inference package
pmtDNA	Primate mitochondrial DNA
PRD	Padded-Recursive-DCM3 decomposition
rCRS	Revised Cambridge reference sequence
RefSeq	Reference sequence
RNA	Ribonucleic acid
rRNA	Ribosomal RNA
SaaS	Software as a service
SMO	Sequential minimal optimization
SRA	Sequence Read Archive
SVD	Singular vector decomposition
SVM	Support vector machine
TN	True negative
TP	True positive
tRNA	Transfer RNA
XSD	XML Schema Definition

One has to be their own biologist.

Daniel M. Gusfield

1 Introduction

The first chapter of this dissertation introduces the bioinformatics' branch on which we have focused throughout this thesis: evolutionary studies, better known as *phylogenetics*. We summarize the terms and concepts we have acknowledge as essential to comprehend the contents of this dissertation. We also present the main challenges on bioinformatics and, more specifically, on phylogenetics. We have divided them in two categories: **i)** the biological-related problems; and, **ii)** the computational challenges, including the problems that arise when we need to process large datasets (*Big Data*). Furthermore, we introduce the biological data we have studied as a real and interesting case for phylogenetic analysis: the mitochondrial DNA. At the end of this chapter, we cover the main objectives that have conducted our research and the organization of this dissertation.

This chapter is divided in six sections. The first one introduces basic biological terms and concepts. The second section covers the biological aspects we have worked on the phylogenetics field. Next, we summarize the computational problems linked with bioinformatics and phylogenetic researches. The fourth section introduces the mitochondrial DNA and its main properties and characteristics. Later, in the fifth section we present the objectives that have directed our research and contributions. Finally, the last section includes the organization of the present dissertation.

1.1 Molecular biology basics

There is no full agreement on the definition of *Bioinformatics*, since researchers do not always agree upon the scope of its use within the biological and computer sciences. We use the most usual definition of bioinformatics under the molecular biology perspective offered by Dr. Tekaia at the Institut Pasteur: "The mathematical, statistical and computing methods that aim to solve biological problems using DNA and amino acid sequences and related information."[1]

In molecular biology, the basic unit of information equivalent to the *bit* in informatics, is the *nucleotide*. There are 5 different nucleotides: Adenine, Guanine, Cytosine, Thymine and Uracil. They are usually represented by their first letter: A, G, C, T and U, respectively. The biological sequences are strings of characters of diverse lengths that are named according to the type of characters they are formed by. There are two types of sequences composed by nucleotides: DNA (***d**eoxyribo**n**ucleic **a**cid*) sequences, which are composed by A, G, C and T characters; and RNA (***r**ibo**n**ucleic **a**cid*) sequences, which consist of A, C, G and U characters.

The DNA sequence carries the genetic instructions of all living organisms and many viruses, containing all the information needed to grow, develop,

[1] *http://www.bioinformatics.org/wiki/Bioinformatics*

function and replicate. This information is arranged in substrings. Depending on the type of information, each substring is categorized as a *gene* or as noncoding DNA. The former is intended to produce *proteins* whilst the latter is related with other functionalities, i.e. replication, regulation, etc. The construction of a protein is a two-step process that involves a transcription and a translation phases. In the transcription, the substring of DNA characters is read and generates its corresponding RNA sequence. Without going into detail, it basically replaces each appearance of T by U. Secondly, the translation stage reads each triplet of characters (better known as *codon*) of the transcribed RNA sequence and replaces them by their corresponding amino acid. Although more than 500 amino acids are known, only 20 appear in the genetic code [1]. This simplification of the initial 64 combinations to only 20 possibilities can be seen as a natural robustness mechanism to keep the proteins, and ,thus, their functionality, intact versus possible errors, i.e. changes (better known as *mutations*) in one or more nucleotides of the gene or the RNA sequence due to diverse factors.

1.2 The *bio* in bioinformatics …

This thesis is focused on phylogenetics, the part of molecular biology directed to evolutionary studies. This area of knowledge studies and tries to comprehend the molecular evolution relationships that exist between organisms or groups of organisms (species, populations) regarding only their biological information, i.e. their DNA, RNA or protein sequences.

The phylogenetic analysis workflow can be depicted as a two-phase procedure: first, we need to determine what the phylogenetic (evolutionary) tree looks like and, afterwards, we can proceed with its study and analysis. The phylogenetic inference is an additive procedure composed by up to four specialized stages. First of all, we need to fetch the raw data we are going to use to build our phylogeny (phylogenetic tree). This stage is usually classified as a pre-workflow stage or stage 0 in many studies, but even so, it is an essential step because not all the sources of information might be specifically targeted for the research we want to perform. The second step preprocess the information to make it suitable for the phylogenetic estimation that will come afterwards. We will dig deeper into these two topics in the following chapters. The third stage is intended to estimate the evolutionary tree. There are several methodologies that will be covered in the fourth chapter. Finally, the last stage can be found in those studies involving different genes or species, where each one is handled independently through the second and third stages. Their results are merged in this last phase obtaining a unique final phylogenetic tree. Since these are also phylogenetic-related methods, we will cover them in the fourth chapter too. Besides, certain biological scenarios necessitate slight modifications of the workflow, repeating specific stages under different parameterizations or skipping them completely.

The latter phase of the phylogenetic analysis workflow comprises analysis methods that have been designed and implemented for different purposes, e.g. graphical phylogenetic displayers, which allows experts on the field to visualize and analyze the given phylogeny [2, 3]; conservation analysis, to study the evolutionary variability of determined positions of biological sequences [4]; etc. We will deepen into these analysis tools in the sixth, seventh and eighth chapters of this dissertation.

1.3 …and the *informatics* perspective

The bioinformatics community grew founded on the challenging problems posed by the molecular biology field, along with the new area of application of existing algorithms developed by computer scientists. Besides, as we will see thereafter, the growth in the amount of available biological data made obsolete most of the hand-methods used by biologists decades ago, requiring the usage of computers to fasten up the procedures.

Nowadays, there are still unsolved problems that are of main interest for both biologists and computer scientists. For instance, try to foresee how a protein sequence will fold in space. The amino acids we introduced earlier have different sizes, diverse chemical properties, some are hydrophilic and others are hydrophobic, . . . Thus, it is not straightforward to predict how a given protein sequence will fold into its 3D structure. Right now, many methods relay on the existing information of other related proteins that have been analyzed to forecast the outcome [5, 6]. Another example is the modeling of metabolic systems. These systems allow researchers to comprehend the set of chemical transformations within the cells of living organisms that keep them alive. The mathematical complexity behind these modelling predictions based on previous biological information, altogether with different basic models (e.g. structural models, gene regulation models, etc.), have posed another challenge in the bioinformatics field [7, 8].

Phylogenetics does not fall apart from the tree regarding challenging and open problems. Adopting the workflow presented in the previous section, one of the first problems we are going to face is the multiple sequence alignment. Many phylogenetic inference methods need their input sequences to have the same length, a requirement that does not always meet. This problem will be discussed in greater detail in the third chapter, but we advance that it has been proven to be $\mathbb{NP}$-*hard* [9]. Moving to the phylogenetic inference problem itself, different methods have been proposed to solve it, but the most relevant of them have also been proven to be $\mathbb{NP}$-*hard*: the maximum parsimony [10] and the maximum likelihood [11]. In all these situations, the application of heuristics has provided an acceptable solution. The relevance of bioinformatics in the informatics community has echoed on the development of programming languages, where many projects were launched in order to create computational biology libraries intended to ease the work on the field [12–14].

1.3.1 Big Data on Molecular Biology

From mid to late 1990s several new methods for DNA digitalization (*sequencing*) were developed. These techniques are more commonly referred in the literature as NGS (***n**ext-**g**eneration **s**equencing*) or high-throughput sequencing [15]. The implementation and commercialization of the aforementioned methods occurred in the early and mid 2000s, producing a huge impact in the sequencing's economic cost and the amount of available digitalized sequences. This agrees with the noticeable change of slope of the lines presented in Figure 1.1.

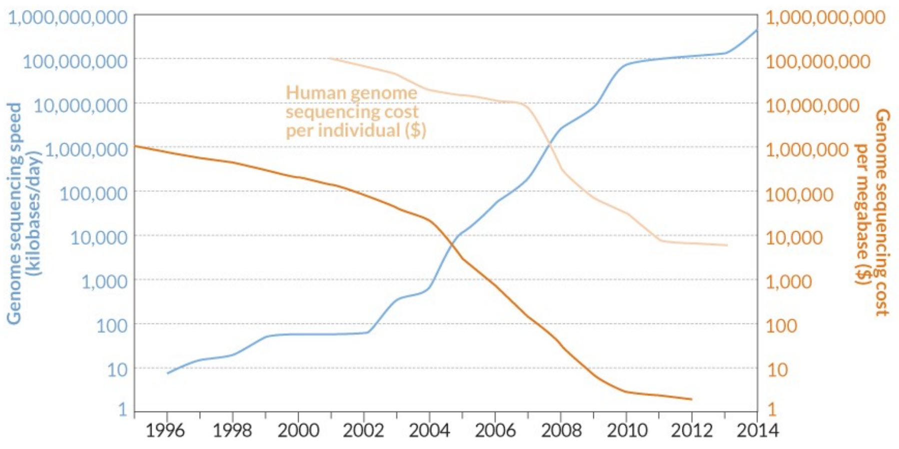


Figure 1.1: Evolution through the years of the sequencing speed (blue line) and the sequencing economic cost (orange line). We can see how the emergence of the NGS changes the slope of both lines in the early and mid 2000s [16].

These innovations changed the scope in molecular biology, embodying the majority of its problems into the Big Data category. For instance, the number of available sequences for the *Homo sapiens sapiens* in GenBank [17], a well-known public database that we will introduce in the next chapter, on 1/Jan/2000 was slightly higher than 180000. On 1/Sep/2016 there were almost 14 million sequences. Most of these sequences have more than 1000 characters in length, and around 60000 sequences are larger than 100000 characters, which is a huge volume of information. Furthermore, it is also interesting to access all the information associated to each sequence, which at least doubles the previous data volume to be processed.

1.4 Special case of study: Mitochondrial DNA

The mitochondrion is a specialized cellular subunit (or *organelle*) found in all eukaryotic organisms which provides most of the energy the cell requires. It is unique in its kind because it is one of the few organelles that have their own DNA, separated from the nDNA (*__n__uclear __DNA__*). As it is shown in Figure 1.2, the mtDNA (*__mit__ochondrial __DNA__*) is a circular double-stranded DNA, composed by 16569 nucleotides on average. It contains 37 genes, 13 of which encode for proteins, 22 for tRNA (*__t__ransfer __RNA__*) and 2 for rRNA (*__r__ibosomal __RNA__*). Besides, there is an important region known as the *Control Region* or *D-loop* (gray segment in Figure 1.2) where the start and ending of the sequence are assembled to form the circular molecule. This segment contains 2 HVRs (*__h__yper__v__ariable __r__egions*) that are very useful for evolutionary and anthropological studies.

The mtDNA has several properties that make it a fascinating candidate for molecular biology and evolution studies. First, in most species the mtDNA has a maternal inheritance [18], which means that the biological information of the mother is transmitted without any alteration to the progeny. This enables genealogical researchers to trace maternal lineage far back in time. Furthermore, its location drives it to a far more vulnerable state than the nDNA, exceeding by a factor of 10 the mtDNA mutation rate in comparison with that of the nDNA [19]. In contrast, the mtDNA has a lower relative degradation [20].

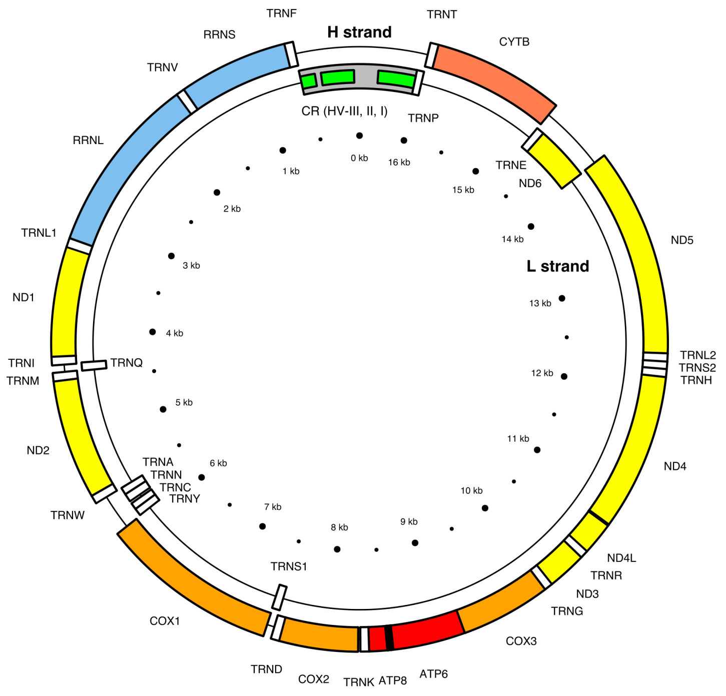


Figure 1.2: Genome of the hmtDNA. The protein encoding genes are marked in yellow, orange and red; the tRNA genes are shown in white; and the Control Region is illustrated in gray. *Picture:* ***Emmanuel Douzery***.

The hmtDNA (***h****uman* ***m****ito****c****hondrial* ***DNA***) was the first significant part of the human genome to be sequenced. A consequence of this accomplishment is the fact that many complete hmtDNA sequences are available. For instance, Figure 1.3 shows the growth of complete hmtDNA sequences in GenBank from 1/Jan/1993 to 1/Sep/2016. As we have aforementioned, an input of more than 32000 strings of 16570 characters on average represents a very large volume of information to handle, requiring in most cases a not negligible amount of computational resources. All the experimental results presented in the following chapters are based on partial or complete mtDNA and hmtDNA sequences.

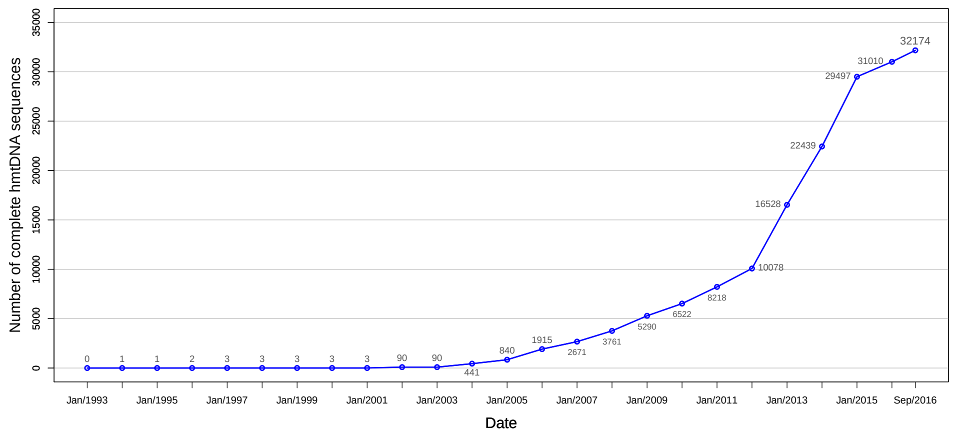


Figure 1.3: Number of complete hmtDNA sequences stored in GenBank from 1/Jan/1993 to 1/Sep/2016. The foundation query used is: *"homo sapiens"[porgn] AND mitochondrion[filter] AND biomol_genomic[PROP] AND "complete genome"[All Fields].*

The applicability of studying the hmtDNA's evolution is based on the relevance of the mutations that may affect its genes. For instance, a short branch in a phylogeny represents a group of individuals that emerged from a mutation (parent node) and died in a few generations or they were sterile. These kind of branches are usually classified as pathogenic, that is, they are related with some known or unknown disease. Most mitochondrial pathogenic mutations are associated to chronic and mortal diseases [21]. Furthermore, the results of several epidemiological studies have addressed that the genetic variation of populations play an important role in mtDNA. This is the case of multifactorial diseases such as Parkinson, Alzheimer, cancer, type 2 diabetes and cardiomyopathy, among others [22].

1.5 Objectives of this PhD thesis

The aim of this thesis is the systematization of phylogenetic methods and software tools to improve the accuracy and performance of phylogenetic analyses compared to their standalone counterparts. The automatic and unattended execution of phylogenetic workflows altogether with inner parallelization techniques, involvement of unbiased biological knowledge and self-setting of proper interaction between methods result in unprecedented improvements on both accuracy and performance of such systems. Besides, we cannot neglect the benefits that a full configuration and broad software tool choices grant in terms of adaptability to user needs. The systematization can rely on a multiplatform framework, disposing of many hardware architectures. Their frequent multi-task capability surpasses by far the delay introduced by the platform's scheduling. We want also to provide solution to the difficulties that arise in large-scale scenarios, where common software tools become inefficient or, even worse, they do not converge. To this end, the introduction of the systematization techniques is summarized under three different perspectives:

1. First of all, our target is to provide a common framework for most phylogenetic studies, where any workflow can be configured and built easily by both computer scientists and biologists. We consider that one of the most unnoticed characteristics that require implementation on molecular evolution workflows is the compatibility between the software tools involved. Although there is at least one file format per type of data that has been standardized, so every software tool that produces them follows the rules settled, not every method can read and output each format available. Therefore, most developers select and fix the methods and parameterizations of the implemented system to assure agreement between outputs and inputs. Alternatively, when that choice is unrealizable, they include ad-hoc format conversion scripts to solve the "incompatibilities". On the other hand, settling the workflow

and its methods at the design level limits the applicability of the resulting system on divergent phylogenetic studies. The systematization of phylogenetic software tools should not jeopardize their flexibility to be applied in different scenarios. Thus, it is essential to provide a solution to each specific need, whenever there is one, granting a broad method selection and all their parameterizations.
There are several systematizations that prove this lack of flexibility, like SATé [23, 24], DACTAL [25] or ZARAMIT [26, 27], which are well-known and used phylogenetic inference systems that have fixed workflows, software tools and parameterizations. We will explore them further later on in this dissertation.

2. The second aspect of our approach is to improve the total performance of phylogenetic inference and analysis systems in comparison to the sum of standalone executions of each procedure concerned. Several phylogenetic inference systems apply divide-and-conquer strategies for biological purposes only. Moreover, parallelization has become a basic technique in molecular evolution software implementations. These premises are a good starting point on behalf of performance improvements. Nowadays, most scientists have access to multi-core hardware platforms, like clusters or clouds, which provide a huge number of resources and affordable execution times for parallelizable tasks. Therefore, to create a successful systematization in terms of efficiency, the process might include three elements: **i)** naïve divide-and-conquer programming techniques when the biological ones are not applicable, and as long as they do not have a negative impact on the accuracy, to generate as many independent parallelizable tasks as cores available; **ii)** task-level parallelization techniques, apart from the instruction-level parallelism aforementioned; and, **iii)** an adequate computational paradigm to correlate the phylogenetic workflow and its implementation for the hardware platforms available. The result-

ing systematization will perform efficiently even on very large-case scenarios.

3. The third and last objective is the enhancement of accuracy of the inferred phylogenies and their analyses with the incorporation of biological-driven methods. Phylogenetic inference systems' design usually starts at the preprocessing stage, that is, the treatment of the biological data necessary to infer the phylogeny. This means that the selection and fetching of the biological information to be processed is made by hand, even if there are some basic software alternatives. For instance, Biopython [13], a biological computation library for Python, includes the *Entrez* module to fetch biological sequences from a set of public databases automatically. As we have claimed before, experts on the field take advantage of their knowledge to choose particular methods for their case of study to improve the accuracy of the workflow developed. For example, a common practice is to perform a gene division of the input biological sequences to study the evolution of each gene separately. Whilst their methods usually rely on heuristics and other approximations that may bias the results, we could design new algorithms that rely only on the metadata available for each biological sequence selected, e.g. the information of the position of each gene provided for most sequences stored in GenBank.

To sum up, this approach allows constructing more flexible and adaptable phylogenetic inference and analysis workflows for most current and future researches. Additionally, the incorporation of clustering and parallelization techniques altogether with multi-core hardware platforms provide a huge enhancement of the performance in most phylogenetic studies. The multiplatform property also grants an automatic and unattended execution after the configuration process and scalability to face the processing of large datasets efficiently. Finally, the biological knowledge gathered by experts on the field is the best resource to develop new specific-purpose software tools. Besides, its incorporation in phylogenetic inference systems increases their accuracy compared to their general-purpose counterparts.

1.6 Organization of this dissertation

The contributions presented in this dissertation have been published in several international conferences and journals. These contents are distributed in nine chapters designed to help the reader to comprehend how phylogenetic trees are inferred from scratch and how they are analyzed afterwards. The biological motivation and background of each topic presented in this dissertation, as well as the conclusions, are covered in their corresponding chapter.

Following this introduction, Chapter 2 summarizes the fetching process required to gather all the biological information we need to perform a phylogenetic study. The information can be biological sequences to build a phylogeny from scratch, a single phylogenetic tree to be analyzed, or two or more related phylogenies that phylogenetists will merge into a single, final tree. Next, Chapter 3 describes the preprocessing stage that usually follows the fetching task to make the data suitable to infer the phylogenetic tree. These preprocessing methods are sequence alignment and dataset division following different criteria, e.g. gene splicing or evolution clustering. Chapter 4 covers five approaches to infer phylogenetic trees, two methods to assemble related phylogenies into a single tree and the criteria used to compare and choose the most accurate phylogenetic tree. Later, the contents of the three previous chapters are joined together to shape phylogenetic inference systems, presented in Chapter 5. Moreover, we describe an alternative solution to keep the phylogenies updated when the appropriate phylogenetic inference process is costly in both time and computational requirements. The next three chapters summarize three molecular evolution analyses, that is, analyses performed by phylogenetists to extract information from inferred phylogenetic trees. Chapter 6 presents visualization software tools aimed to display phylogenies and to provide specific features to study them. Following, Chapter 7 covers the study of the conservation of every possible value at each position of a set of biological sequences. These results help to determine functionality-related and critical positions, that is, sites where a mutation might trigger a disease. Chapter 8 introduces prediction software tools created to classify mutations by their pathogenity. They base their decision on diverse criteria, like the conservation analysis aforementioned, to determine if a mutation is most likely associated with a disease or not. Finally, Chapter 9 gathers the conclusions drawn from this research and outlines the future work.

I do not fear computers. I fear the lack of them.

Isaac Asimov

2
Sequence fetching

Once a biological sequence has been digitalized, it is important to store it and all its related information, e.g. source organism or papers in which it has been studied, so that the expensive sequencing process is run only once for the same sequence. This idea encouraged the creation of many biological databases. There are diverse aspects that classify them in different categories like their kind of access (public or private), the type of information they store (only DNA or mixed sequences) or the interface they offer (via web, SQL queries, . . .), among others.

In this chapter we introduce MEvoLib, the first molecular evolution library for Python [28]. It takes advantage of some classes and modules implemented in Biopython [13], one of the most extended and used computational molecular biology libraries for Python. MEvoLib has been designed to provide integrity with other tools and scripts that use Biopython, as well as to avoid the replication of already functional and public modules. We found two important shortcomings in Biopython that encouraged the creation of MEvoLib. The first one is the lack of flexibility towards supported software tools. For instance, MAFFT [29] (a multiple sequence alignment tool that will be introduced in Chapter 3) has a dedicated module with a fixed parameter list. The main problem is that not every parameter has a 1-to-1 relationship with those of the software tool, producing a negative repercussion on its learning curve. The second shortcoming is its scope. Biopython lacks of support for many important software tools and methods required in molecular evolution studies. MEvoLib meets this requirement, establishing a symbiosis with Biopython. Since MEvoLib offers several interfaces related with the processes involved in common molecular evolution studies, each one will be deeply covered in their corresponding chapters.

This chapter is divided in four sections. The first one introduces several molecular biology databases and their most relevant properties. The second section covers the common fetching procedures already available. In the third section, we present the MEvoLib's module we have designed to fetch automatically the desired biological information, providing some improvements to the current solutions. This contribution has been published as part of the journal paper [28]. Finally, the last section gathers the conclusions of the work presented in this chapter.

2.1 Molecular biology databases

Multitude of molecular biology databases have been created regarding different purposes [30]. Moreover, most of them are publicly available. One of the most important and accessed databases for DNA and RNA sequences that follows this criterion is GenBank [17], which belongs to the NCBI (_**N**ational **C**enter for **B**iotechnology **I**nformation_) [31] along with other databases. Another relevant example is BLAST (_**B**asic **L**ocal **A**lignment **S**earch **T**ool_) [32], which provides a similarity search were all the sequences related to a given one are returned. It provides different databases for specific types of information (nucleotide, protein, ...). Finally, MITOMAP [33, 34] poses as a specific purpose example, being a database that stores only the human mitochondrial genetic information.

Besides the huge amount of sequences stored, GenBank offers a lot of associated information for each sequence, including its source (e.g. organism, region), the gene and non-coding DNA splicing, and other relevant information. It has been designed with a user-friendly website access [17] and supports a query-like search of the desired information. GenBank has included unattended access given the query and other relevant information through scripting. The user can also select whether to download the raw sequences or include all their associated information in different text formats too. The same criteria have been followed by other NCBI databases, facilitating the development of scripts that can retrieve all the information automatically.

2.2 Related work

It is common to read in many bioinformatics papers detailed information about the datasets they are going to use for their research and the source database chosen, but just a few mention how those sequences were obtained. This is because they are retrieved manually in most cases, that is, the person in charge of that task connects to the database's web site, selects the desired sequences and downloads them. To such task, NCBI developed Entrez [35], a primary text search and retrieval system that integrates most of its databases. It provides a comprehensible interface for biologists to use, translating the high-level query to the inner database's language.

With the incorporation of scripts and software tools as regular resources, this query system led to the creation of eUtils (***E**ntrez Programming **Util**ities*), a set of server-side programs that provide a stable and structured interface into Entrez. They use a fixed URL syntax that translates a standard set of input parameters into the values necessary for various NCBI software components to search for and retrieve the requested data. Thus, any programming language capable of sending a URL to the eUtils server and interpret the XML response can be set as framework to develop a fetching program. This is the case of the *Entrez* interface provided by Biopython.

There are other software tools developed to retrieve biological information from public datasets. The SRA (***S**equence **R**ead **A**rchive*) toolkit [36] is a set of binaries available for frequent operative systems that fetch the data submitted into the public repository of primary next-generation sequence data archive.

2.3 *MEvoLib.Fetch* interface

The first module of MEvoLib we present in this chapter has been implemented to fetch biological information, either from local files or from different NCBI databases. This module has been divided in two classes: *BioSeqs*, to fetch biological sequences from both sources; and *PhyTrees*, which fetches phylogenetic trees only from local files[1]. Both classes can work with any file format supported by Biopython. Once all the desired information is gathered, it can be written in a GENBANK or NEWICK file format, respectively, along with a report file, which is saved with "*.rep*" extension and the same filename. This last file encloses the number of sequences or trees stored, together with a detailed list of all the data sources, the date and time when they were accessed and the fetching instructions used, i.e. complete path of the local file or the NCBI database and query.

[1]There is, to our knowledge, no public database containing useful or relevant phylogenies to be used in further molecular evolution studies.

The report file strengthens experiment reproducibility, not only storing all the sources' information, but also with a temporal line that keeps track of any data overriding due to an identifier collision from two or more sources. Moreover, the report file can substitute the data file when sharing or publishing research files, especially when the data file is large and none of the sources are a local file.

MEvoLib's *BioSeqs* implements three useful properties to fetch sequences from NCBI databases (e.g. GenBank). The first one consists of a size limitation on the number of sequences fetched for the given database and query. This characteristic is intended for particular researches where the user might just want to retrieve a representative sample. The second feature enhances the download process of large datasets, computing the adequate batch size for the expected sequence's size (in bytes). This calculation is based on the directions given by the NCBI itself, together with a prefixed maximum download time for each batch to optimize the interval and number of requests made. This upper bound also reduces the data loss when an unexpected communication error happens. The third property enhances the update process, removing those sequences that are no longer stored in the corresponding NCBI database, and fetching only new or updated sequences, based on their version.

The large amount of public and accessible information has a main drawback: it is not completely accurate neither useful. After the sequencing process, the researchers are the ones responsible for uploading the biological sequence to GenBank. They are also in charge of including the sequence's metadata, making it error-prone. Despite all the possible revision processes included by GenBank once the sequence is uploaded, we have found many cases that do not follow the established rules and may lead to errors or inaccurate results in further studies. In Appendix B we have included the analysis we have performed about this issue for the hmtDNA.

2.4 Conclusions

In this chapter, we have presented the first interface of MEvoLib intended to ease the fetching of biological information from local files and public databases. Whilst it keeps the best properties of some current approximations, like the automatic fetching, our solution improves those methods in several ways: it can merge data from more than one source and it generates a report file with the relevant information gathered during the fetching process. Furthermore, it adds a batch-fetching handler and an update feature to the *Entrez* module provided by Biopython, enhancing the database interaction procedure.

As future work, we contemplate to reflect in the report file those manual modifications done by the user to the fetched dataset, e.g. deleting a specific subset of sequences. Besides, the current implementation stores all the information of the sequences/trees in memory, which can have a negative impact on the performance of our library. In the worst-case scenario, an exception might be raised if the total amount of information to fetch exceeds the amount of memory available, losing all the downloaded data. Thus, a new file-oriented design of this interface will be considered for its next version. This rearrangement of storage will slightly slow down the method's performance but it will remarkably increase the amount of manageable data.

He who knows when he can fight and when he cannot,
will be victorious.

Sun Tzu

3 Sequence preprocessing

Raw biological sequences are usually inaccurate, comprising a large number of uncertainties that make them unsuitable for knowledge extraction before an adequate preprocessing. Thus, many bioinformatics procedures directly expect a normalized input so that they can perform their estimations. For instance, if we would like to study the evolution process of a particular gene in several organisms, it is certain that some of those gene sequences will not have the same length or they may vary in some character values for corresponding positions. These discrepancies are even more likely to appear if the target organisms belong to different species.

Moreover, the computational resources required to handle the massive quantity of molecular sequences available in many cases are often unavailable for various research groups. Nevertheless, there are well known computational techniques like divide-and-conquer and clustering, among others, that can usually be applied in these scenarios and provide a near-optimal solution.

The first section of this chapter introduces the biological concepts and ideas, which justify the usage of preprocessing methods. The second section is dedicated to the multiple sequence alignment, a preprocessing procedure designed to even up the length of all the biological sequences. We include the state-of-the-art of the topic and our contribution with one of MEvoLib' modules. The third section covers the application of diverse computational techniques to more biological-oriented preprocessing tasks, including four interfaces of MEvoLib we have developed for general and specific scenarios. All our contributions have been published as part of the journal paper [28]. Finally, the last section assembles the conclusions for this chapter.

3.1 Biological motivation

The biological sequences can suffer changes (*mutations*) in their information due to several factors. The most common risk situation is when these sequences are synthesized in their natural environment, that is, within the cells. DNA sequences and, in some cases, RNA sequences as well, are duplicated when the cell is about the divide in order to generate two exact copies from the original one, i.e. the replication process. But this process is not infallible: the error rate is of the order of 1 mistake per 10^7 or 10^8 nucleotides [37]. Despite not being a substantial rate, we have to take into account that DNA sequences are usually really large (about 6×10^9 nucleotides in the case of the human nDNA). This means that, in the worst-case scenario, around 600 mistakes can be made each time the cell replicates. Fortunately, there are some cellular mechanisms, namely the proofreading and the mismatch repair, which look for these mistakes and fix them, raising the error correction rate up to 99%.

The RNA is usually synthesized from a substring of the DNA, a gene, through the transcription process we introduced in the first chapter. “The error rate of RNA synthesis is of the order of one mistake per 10^4 or 10^5 nucleotides ... The much lower fidelity of RNA synthesis can be tolerated because mistakes are not transmitted to progeny. For most genes, many RNA transcripts are synthesized; a few defective transcripts are unlikely to be harmful” [38]. The translation of each RNA sequence synthesizes the encoded protein. The study of this process showed an error rate of about an order of magnitude higher than those in transcription, that is, 1 mistake per 10^3 or 10^4 amino acids. Each amino acid is encoded by three nucleotides (*codon*), so the length of protein sequences is rather smaller than that of DNA sequences, which usually include several genes. This suggests about one error per 10 proteins formed for a common length protein ($\approx$ 300 amino acids) [39].

The metabolic activities and other environmental factors like radiation can also have a negative impact on the biological sequences, producing mutations. For the former, the cells have repairing mechanisms to neutralize them given their nature and frequency. For the latter, these repairing mechanisms might not be enough, leading to major damages or even the death of the cell.

Whilst we could name all the previous cases as *natural* mutations, there are others that have to be classified as *artificial*. We are referring to those introduced by the sequencing methods. The digitalization process of biological sequences has been improved over the last decades, but even so, it is hard to get an accuracy higher than 99.999% [40]. Even a 0.001% of error rate can have a huge impact in the resultant sequence if we are handling large DNA sequences, like the human nDNA aforementioned.

How do all these mutations affect the biological sequence? They can generate changes in one or more values of the sequence, leading to insertions or deletions of one or more characters, known as *indels*, and there are some rare cases where a whole substring changes its position (*locus*) to another part of the biological sequence, known as *translocations*. The former keeps the length of the biological sequence intact, but they are really important due to their fitness to produce changes in the resultant protein when they affect coding genes, manifesting as a disease. Indels are relevant due to their modification of the biological sequence' length, in addition to the possible effects on the coding regions. Translocations hinder genetic studies and the establishment of evolution relationships among different organisms or species. The complexity of this natural phenomenon forces us to run string-matching algorithms to assure that a gene of one organism has been removed in another one's sequence.

Therefore, researchers need to arrange DNA, RNA or protein sequences in a comparable way if they want to identify regions of similarity that may be a consequence of structural, functional, or evolutionary relationships. That arrangement is called alignment, and it tries to set as many identical or similar characters as possible in successive columns. The alignment process introduces a new character, the *gap* (usually represented by the symbol "–"), which is introduced in between characters to adjust the length discrepancies between sequences. As consequence, all the sequences will end up having the same length.

As we have aforementioned, not every mutation has the same impact on the organism. In the same way, these changes might or might not be inherited by their offspring. Therefore, different genes of the same organism might display different evolving pathways. Besides, the environment and origin of diverse individuals within a specie have also a means to affect their genetic information. Thus, it is a common practice to work with each gene from the same set of organisms independently, or handle groups of individuals separately regarding certain biological conditions, especially in phylogenetic studies. The *haplotyping* is a good example of the latter. The haplotype refers to a specific position within a DNA sequence that has been tagged as biologically relevant. A mutation at this site is usually depicted as a new branch in the phylogeny for that type of sequence, as it happens with the hmtDNA [41]. The clusters generated applying this biological knowledge are named *haplogroups*.

3.2 Sequence alignment

In this section, we cover the alignment process to amend the existing differences between raw biological sequences. First, we will cover the state-of-the-art about alignment software tools. Next, we will present the work we have performed to provide a new solution to integrate those methods in our programs with MEvoLib.

3.2.1 Background

The computational complexity of obtaining the optimal solution for an alignment of n biological sequences of length m is $\mathrm{O}\left(m^n\right)$ [42] making it forbidding even for a small-sized problem. An alignment of 2 sequences is usually referred as a *pairwise* alignment, whilst if more than 2 sequences are involved it is named a MSA (***m**ultiple **s**equence **a**lignment*). Furthermore, the MSA problem has been proven to be $\mathbb{NP}$-*hard* [9, 43, 44]. There are just a few scenarios where scientists will need to compute pairwise alignments, thus we will focus the contents of this section on MSAs. The complexity statement has led to the creation of many heuristic algorithms to settle a good correlation between the accuracy of the resultant MSA and its computational cost.

Multiple software tools have been developed to cope with the MSA problem. Some of the most used and well-known are MAFFT [29], Clustal Omega [45], MUSCLE [46], KAlign [47] and PRANK [48]. Despite providing different approximations to achieve the alignment of the input sequences as fast as possible, the biggest difference among these methods is their *score* function. The *score* is a numerical number given to an alignment symbolizing its proximity to the true alignment, that is, the alignment that would perfectly match with the events (mutations) occurred to each sequence. The score function is a mathematical system composed by one or more *penalty* equations whose variables usually rely in three main factors: the mismatching of characters (e.g. putting an A and a T in the same column for two different sequences), the insertion of a new gap, and the number of consecutive gaps introduced in the same sequence. The gap is, as we have aforementioned, a useful resource to adjust the discrepancies between sequences but it must be handled with caution: setting one or more gaps in an area where there was not an indel can lead to an erroneous analysis of the data. Moreover, the methods' accuracy can vary significantly regarding the size of the input dataset, the type of its biological sequences and their similarity.

All the aforementioned MSA tools have been designed to provide a command line interface, which can be harsh for some non-computer users. Thus, there are some alternatives based on a visual interface, like Jalview [49] or SeaView [50]. We can also find MSA software systems like PASTA [51], which rely on an accurate evolutionary tree to perform the alignment procedure. Additionally, some software libraries offer MSA tool modules to integrate the alignment process in other programs or scripts. For instance, Biopython [13] has implemented specific interfaces for MAFFT, Clustal Omega, MUSCLE and PRANK, among others.

3.2.2 *MEvoLib.Align* interface

MEvoLib includes the *Align* interface to work with different MSA tools. Its creation was encouraged by the lack of flexibility of the MSA interfaces offered by Biopython. For instance, we have already mentioned in the previous chapter problems with the parameter list for calling MAFFT: it is fixed and not completely related with the arguments allowed by the software tool. Additionally, the recent published updates have not changed along with the new versions of the software, so the new functionalities are not supported (e.g. `--add` feature introduced in version 7.273). Thus, the user must learn and comprehend both interfaces in order to handle properly the MSA tool through Biopython.

The main characteristic of MEvoLib.Align's interface is that there is only one parameterization for every MSA tool included. The user has to indicate the tool to launch, the input sequence file, the file format and the list of arguments of the tool. The resultant alignment will be returned in a *MultipleSeqAlignment* Biopython object. In addition, an output file and its format can be added to the previous list of parameters in order to save the resultant MSA. The purpose of the input and output file formats is twofold: **i)** to ensure that we can read and write in those formats (through Biopython); and, **ii)** to apply format conversion if the input or output formats are not supported by the MSA tool. The latter facilitates the work of the user and it improves the interface's usefulness in scenarios where the next process requires a specific file format.

For each MSA tool included in MEvoLib, we have created a configuration dictionary. It contains common parameterizations referenced by a keyword that can be passed to the *Align* interface instead of the list of arguments. This addition improves the readability of the code and should facilitate recalling the whole set of configurations. The user can also append other keywords and configurations in the corresponding MSA tool file of the library. We have incorporated in every dictionary a *default* keyword, which will be used if no configuration parameters are provided. It is also expected to ease the initial work of novel researchers.

In the first version of our interface, we have included MAFFT, Clustal Omega and MUSCLE. Moreover, the method can launch MSA tools not contemplated in the installed version of the library. To do so, the user must include two new elements into the previous parameter list: a list of supported input file formats and the input file command (e.g. `-in`). However, this functionality is intended for expert users only.

3.3 Partitioning and clustering

The next section summarizes the preprocessing procedures related with the division of the input dataset into smaller subsets. After covering the background on this topic, we introduce the two main clustering criteria: **i)** apply a divide-and-conquer strategy in this stage to enforce parallelization techniques in the following procedures; **ii)** take advantage of the biological knowledge to improve the accuracy of next stages by dividing the data into more biologically related subsets. In both cases we have contributed with novel approaches included in MEvoLib's *Cluster* interface. Each one of the four different approaches implemented return the clustering as a dictionary, with set identifiers as keys and one list of *SeqRecord* Biopython objects per key as values.

3.3.1 Background

Divide-and-conquer and parallelization techniques are regular choices in the toolset of every informatician, even more when dealing with large-case scenarios or lengthy procedures. In bioinformatics, they are used frequently given the recurrence of the problems meeting one or both of those scenarios. There are, to our knowledge, no published implementations for a naïve, non-biological, division of sequence datasets. We are certain that many studies have performed such practice but they might have been done it by hand or by scripts that are not provided altogether with the paper.

As we have stated before, there are some cases where the sequences can be divided or clustered under biological fundaments. The results obtained under these premises are often more meaningful and relevant for the biology community: they can validate previous analyses and/or establish new hypotheses. In this dissertation, we will be concerned about two of those fundaments, but we acknowledge there exist others. The first one resorts to the evolutionary information reflected in a phylogenetic tree. The sequences are clustered into subsets that represent relatively close sets of evolution events. We can find diverse software tools that follow this principle, like Rec-I-DCM3 [52] or its variation named PRD (***p**added-**R**ecursive-**DCM3** decomposition*) [25], that use the phylogenetic tree to generate overlapping subsets regardless of the type of biological information they contain. A method in this stage that requires a phylogeny as input states an inevitable infinite loop in the phylogenetic workflow we are picturing throughout these chapters. We will discuss this problem on chapter 5. The second fundament takes advantage of the biological knowledge we have about the specific type of biological sequences given as input. For instance, the haplogroup division of a set of complete hmtDNA sequences can be achieved by Phy-Mer [53] or Haplo-Grep2 [54]. As to the gene partitioning, the only technique we are aware of is the pairwise alignment of each sequence of the input dataset with their reference sequence, that is, a sequence that represents a canon for that specific type of biological sequence. Afterwards, we can use the gene locus of the reference sequence to extract the genes from the input sequence. We have not found any implementation of this or any other similar technique.

3.3.2 Naïve approximation

The first approximation is a naïve mathematical approach to the division problem. We process the input dataset as a $n \times m$ matrix of characters. Given the possible discrepancies in length, we will behave as if white space characters (“ ”) where added at the end of each sequence to meet the length of the longest one in the dataset.

Now we can reduce our problem into smaller subproblems dividing the given matrix by rows, columns or both. MEvoLib covers both concepts with two methods: *naïve rows* and *naïve columns*. The former divides the input sequences into the given number k of disjoint sets. To get the most balanced possible output, the method will generate $j = n \bmod k$ sets with $\lceil {}^{n}/_{k} \rceil$ sequences and $k - j$ sets with $\lfloor {}^{n}/_{k} \rfloor$ sequences. The latter calculates the fragment length and the range indices based on the longest input sequence m and the given number of sets k. The algorithm applies the same equations as the previous method, replacing n by m.

3.3.3 Integration of biological knowledge

As we have claimed before, the experimental viability of the naïve clustering relies on the homogeneity of the biological sequences and on the following procedures. In other words, those sequences that may improve the accuracy of the results by being clustered in a determined way, either by rows or by columns, are not suitable for the naïve clustering methods. For instance, mixing parts of one gene with another (that is, performing a naïve division by columns) can lead to an inaccurate phylogenetic tree. Thus, the application of biological knowledge can substantially improve the accuracy of the experiments. On the other hand, the resultant clustering may affect negatively the performance in comparison with that of the naïve approach: whilst with the naïve methods we can match the number of subsets to the number of cores/threads available, with the biological approximation we might end up with an inadequate number of subsets for a straightforward allocation. Additionally, naïve approaches generate roughly equally sized subsets that have almost the same time and memory costs, while the biological approximation is most likely to obtain an unbalance load. Next, we introduce two methods of MEvoLib designed to embed different biological knowledge into the clustering process.

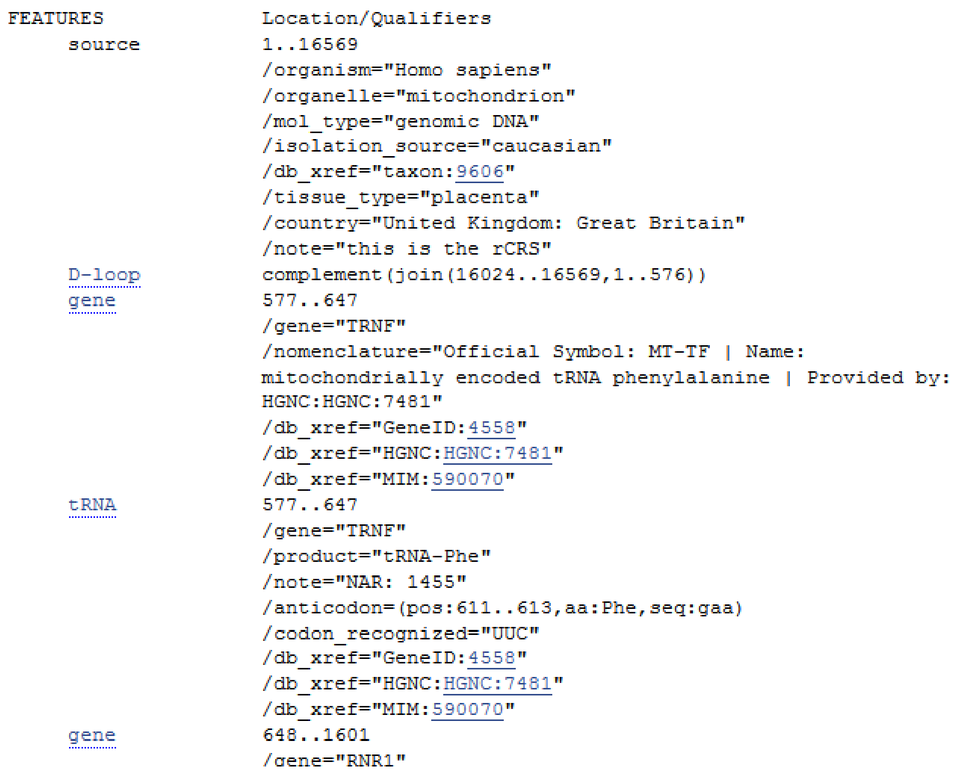
MEvoLib.Cluster's PRD method

Generating subsets of evolutionary related sequences usually requires an input phylogeny containing, at least, the same sequences of the input dataset. Since that is the only requirement, this methodology is applicable for any type of biological sequence. One of the most extended and used methods in this category is the PRD decomposition, a main component of the DACTAL phylogenetic system [25]. It extracts a set of overlapping subsets from the phylogeny, where the union of these subsets equals the input dataset. The method's input is the dataset, the phylogeny, the maximum size of each subset and the number of overlapping sequences. Each subset that does not meet the size condition is used again as input, with its corresponding subtree, for the PRD decomposition process. MEvoLib's *Cluster* interface includes its own Python implementation of the PRD decomposition algorithm (instead of Pearl). Moreover, we have updated its dependencies. The method has been improved regarding its original implementation by replacing its recursive part by a parallelized iterative procedure, where a queue manages the sets that require further decomposition.

MEvoLib.Cluster's Genes method

The genes partitioning is the biological approach to the column division for DNA sequences. MEvoLib incorporates the *Genes* method, designed to generate as many sets as genes available in the input sequences. The main idea is to take advantage of the biological knowledge to extract all the genes of a set of sequences. NCBI databases store this biological knowledge as metadata in each sequence, following a XML-like representation. An example is shown in Figure 3.1. This metadata altogether with the sequence is usually stored in a file format named GENBANK. Thus, we have created a small database with all the available *features* (tags) at NCBI databases and their *qualifiers* (set of valid attributes for that tag). The method admits a feature filter, that is, a list of tag identifiers, as an extra parameter to extract only those features in which the user is interested.

The *Genes* method uses the biological information available in the input file (GENBANK format) to determine the gene loci (location) of each sequence. If there is no metadata available for a specific sequence or the input file is in an information-less format (like FASTA), the method applies



```
FEATURES             Location/Qualifiers
     source          1..16569
                     /organism="Homo sapiens"
                     /organelle="mitochondrion"
                     /mol_type="genomic DNA"
                     /isolation_source="caucasian"
                     /db_xref="taxon:9606"
                     /tissue_type="placenta"
                     /country="United Kingdom: Great Britain"
                     /note="this is the rCRS"
     D-loop          complement(join(16024..16569,1..576))
     gene            577..647
                     /gene="TRNF"
                     /nomenclature="Official Symbol: MT-TF | Name:
                     mitochondrially encoded tRNA phenylalanine | Provided by:
                     HGNC:HGNC:7481"
                     /db_xref="GeneID:4558"
                     /db_xref="HGNC:HGNC:7481"
                     /db_xref="MIM:590070"
     tRNA            577..647
                     /gene="TRNF"
                     /product="tRNA-Phe"
                     /note="NAR: 1455"
                     /anticodon=(pos:611..613,aa:Phe,seq:gaa)
                     /codon_recognized="UUC"
                     /db_xref="GeneID:4558"
                     /db_xref="HGNC:HGNC:7481"
                     /db_xref="MIM:590070"
     gene            648..1601
                     /gene="RNR1"
```

Figure 3.1: Example of how the sequence's metadata is displayed in GenBank.

the algorithm we have introduced in Section 3.3.1. Remember that this algorithm deduces each gene locus from the corresponding RefSeq (***ref**erence* ***seq**uence*). The RefSeq is a sequence that has been checked and validated many times by diverse biologists through time so it is considered 100% accurate and representative of those sequences from the same source, e.g. the rCRS (***r**evised* ***C**ambridge* ***R**eference* ***S**equence*) is the RefSeq of the hmtDNA [55]. We use the corresponding RefSeq for the input dataset, indicated by the user, to generate a pairwise alignment of each information-less sequence with the RefSeq. After that, the method removes the sites corresponding to new gaps introduced in the RefSeq for both sequences and, finally, it applies the gene locus information of the RefSeq to the new sequence. MEvoLib already includes certain reference sequences and their related information. Hence, we avoid unnecessary internet connections to retrieve the data at runtime, preventing potential errors and hastening the methods.

The *Genes* method uses only the biological knowledge provided by the user in the sequences file, preventing any guided or prior information in the source code, apart from misspelling checkers. Thus, the method is very likely to run accurately even if the information changes or the classification criteria are modified due to new discoveries. We wanted to extend the usage of the *Genes* method to expose possible errors in the information available: NCBI databases, like GenBank, do not perform any standardization over the values of each qualifier and feature of each new sequence, generating several nomenclatures for the same gene. Hence, the method implements a merge of algebra of sets and statistical sampling to match terms referring to the same gene, and detect those related terms with a low sampling representation. As a result, a log file is created with each uncertain pair of terms and a list with the sequences that support them. Hence, the user can easily detect sequences that merge two genes due to an error in their information. Besides, this methodology works properly even though the genes might be in different loci, e.g. when the input dataset is composed by complete sequences and fragments.

There is another innovation in the clustering procedure: whilst the classical approach withdraws different genes based only on the information from *gene* or *product* qualifiers of the features provided, the *Genes* method mine the data from every qualifier available. To demonstrate the improvement that the *Genes* method offers in comparison to the classical approach, we downloaded all the complete hmtDNA sequences available at GenBank on 07/Jul/2016. The 31755 sequences used match the following query: *"homo sapiens"[porgn] AND mitochondrion[Filter] NOT mRNA[Filter] AND "complete genome"*. We also created three subsets with the first 100, 1000 and

10000 sequences of the downloaded data to perform an additional scalability test for the method.

Before running all the tests, we executed the *Genes* method with all the sequences in its default configuration in order to analyze the log file to detect possible errors in the biological information. Three sequences were removed from the downloaded set: **i)** KP702293.1 has the gene ND3's product as "NADH dehydrogenase subunit 2"; **ii)** FR695060.1 has the CDS "ATP synthase 6" identified as ATP8; and, **iii)** DQ862537.1 has the gene ATP6's product as "cytochrome c oxidase subunit III". The three subsets were updated too.

To perform the classical approach, we modified the default behavior of the *Genes* method: instead of collecting the information of all the qualifiers of each feature, it only took into account the *gene* or *product* qualifiers, respectively. All the results can be found in Table 3.1. We focused our conclusions on the recovery rate of the CDS (*coding DNA sequence*), rRNA and tRNA features for each configuration. The hmtDNA contains 13 CDS, 2 rRNA and 22 tRNA genes.

Table 3.1 shows that the time and memory usage of the *Genes* method is slightly higher than classical approaches in most cases, but the difference is negligible in comparison to the improvement in the recovery rates. The rRNA genes are always clustered accurately, even when the number of sequences is increased. The CDS genes are correctly gathered when the number of sequences is high enough. For instance, for the 100 dataset, the variety of information available is not large enough and, thus, the *Genes* method obtains 15 genes instead of 13 because it finds two nomenclatures for 2 different genes: "atp6" and "atpase 6", and "atp8" and "atpase 8". The tRNA feature is never recovered completely due to a duplication of two tRNAs in the hmtDNA[1].

[1]The tRNAs for Leucine and Serine have two loci instead of one, located at different sites and strands of the hmtDNA. In just a few cases these two loci are differentiated by a "1" or "2" added to their qualifiers.

Table 3.1: Performance and feature recovery of a classical approach versus MEvoLib's *Genes* method for four sets of hmtDNA sequences. The second column of each feature (*feat.*) shows the result's divergence from the expected value.

Num. Seqs.	Configurations	Time (s)	Mem. (MB)	CDS feat.		rRNA feat.		tRNA feat.	
100	gene	1.56	32.37	26	(+13)	3	(+1)	21	(-1)
100	product	1.41	32.33	18	(+5)	2	(0)	20	(-2)
100	all	1.42	32.46	15	(+2)	2	(0)	20	(-2)
1000	gene	13.15	91.42	26	(+13)	5	(+3)	43	(+21)
1000	product	12.71	92.06	22	(+9)	4	(+2)	20	(-2)
1000	all	13.74	92.63	15	(+2)	2	(0)	20	(-2)
10000	gene	127.95	687.63	31	(+18)	5	(+3)	45	(+23)
10000	product	171.80	692.59	34	(+21)	4	(+2)	20	(-2)
10000	all	136.91	700.73	13	(0)	2	(0)	20	(-2)
31752	gene	412.22	2126.83	33	(+20)	5	(+3)	46	(+24)
31752	product	467.78	2144.32	51	(+38)	6	(+4)	20	(-2)
31752	all	509.78	2177.28	13	(0)	2	(0)	20	(-2)

3.4 Conclusions

In this chapter, we have presented two MEvoLib modules destined for data preprocessing. The *Align* interface provides a single method to execute MSA software tools. In order to ease its usage, we have included a single list of parameters for all the tools included in the current version of the library. A dictionary with common configurations is provided for each tool, including a *default* parameterization too. These properties enhance code readability and help novel users with the first steps. Furthermore, the interface can handle other MSA tools by appending two extra parameters. However, this functionality is intended for expert users only. Additionally, we have implemented an internal format conversion to facilitate its interaction with other modules and software tools. MEvoLib incorporates the *Cluster* interface too, intended to perform the division of a set of biological sequences into subsets under different criteria. It provides two methods for the division into subsets of complete sequences or into fragments as a naïve approach to improve the time cost of following procedures using parallelization techniques and architectures. Additionally, MevoLib includes two methods aimed to take advantage of the currently available biological knowledge. The PRD decomposition method we have implemented is more efficient than its previous version due to a modification on its algorithm: we have replaced its recursive design by a more efficient iterative schema that has been parallelized afterwards. Moreover, the *Genes* method is, to our knowledge, the first gene division algorithm that relies only on the metadata of the input biological sequence to perform its task. The incorporation of algebra of sets and population statistics provides an accurate gathering solution for the high number of existing terms referring to the same gene.

As future work, we plan to extend the MSA tool list included in the *Align* interface of MEvoLib with software tools like PRANK or KAlign. We also contemplate to incorporate new interfaces for the haplogroup clustering software tools presented in the Background section. Furthermore, we aim to improve the efficiency of the methods that have not been parallelized yet to reduce their time cost in very large scenarios.

Evolution is the fundamental idea in all of life science - in all of biology.

Bill Nye

4 Phylogenetic tree production

Once the fetching and preprocessing stages have been completed, the data is ready for the phylogenetic tree inference. The procedure will be different in the cases when the dataset has been kept as a whole, and when it has been divided into subsets. These subsets can correspond to the same or diverse species, or far apart relatives under an evolutionary perspective. The literature refers to the result of the last scenario as *supertree*. The methodologies summarized in this chapter provide different perspectives on how the evolution process affects organisms through time, being a topic still discussed nowadays in the biological community.

In the first section of this chapter, we cover the biological aspects that set the base for the methodologies and software tools presented in the second section. The third section covers two MEvoLib interfaces we have developed to cope with the phylogenetic inference and assembling procedures. This contribution has been published as part of the journal paper [28]. The fourth section presents the machine learning approximation we have worked on as an alternative for the phylogenetic inference process. Finally, the last section assembles the conclusions for this chapter.

4.1 Biological motivation

In genetics, a mutation can be classified as non-neutral, which includes harmful and advantageous ones; or neutral, based on its effect on fitness. According to the neutral model of molecular evolution, harmful mutations, also known as deleterious, are removed by negative selection while those classified as neutral are kept. Advantageous mutations occur so rarely that they can be ignored. As we have aforementioned, the phylogenetic tree is the representation of the evolutionary relationships between various organisms. Phylogenies help researchers not only to understand the past, but also to predict the function of different genes, their interaction, the source and transmission of diseases, and the origin and spread of different individuals or species, among other things. A phylogeny is usually modeled as a rooted binary tree, where each node means an evolution event where a mutation or a set of mutations occurred to one or more organisms, generating a bifurcation on their evolution. The root is considered the common ancestor of all the organisms represented at the tree. The biological information of ancestors is scarce in nature. Therefore, the biological sequences at the dataset are usually located at the leaves of the phylogeny, requiring an inference process to obtain the sequences at the root and inner nodes. Figure 4.1 shows an example of a phylogeny of species. In this case, each node represents the turning point where a group of individuals is no longer considered as one single species, but two.

There are two main kinds of phylogenetic trees regarding their branches. One uses the branches and their length just to symbolize the evolution path of one or more organisms, and it usually displays all the leaves of the tree at the

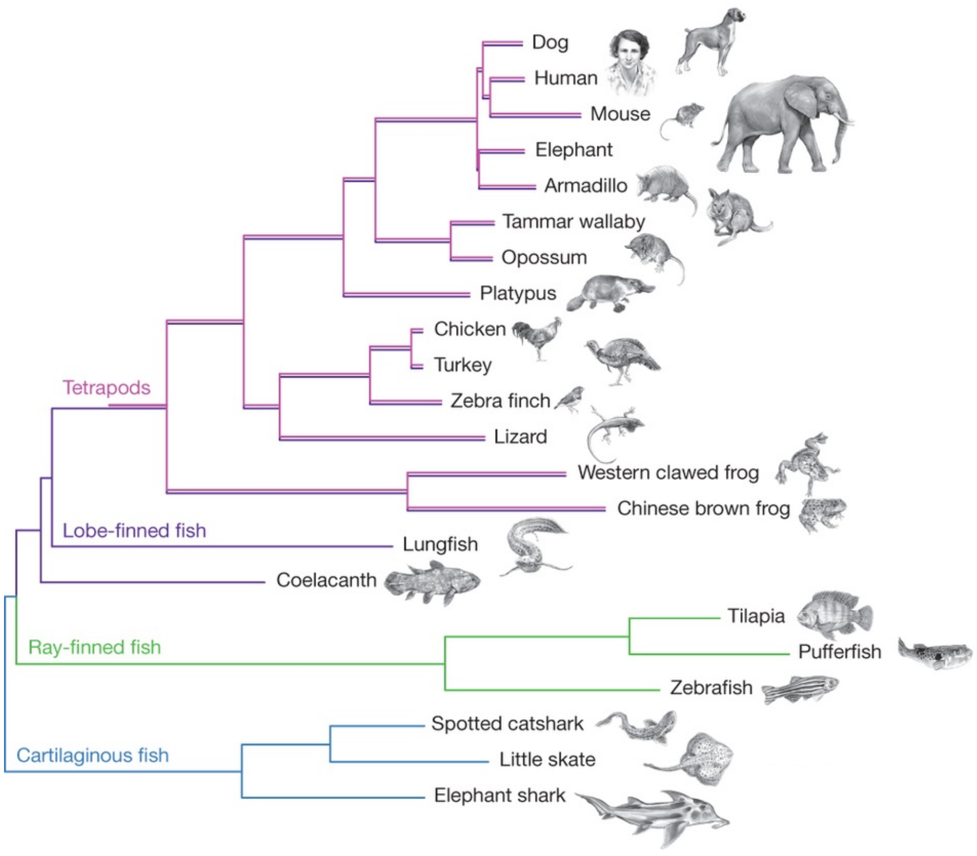


Figure 4.1: Example of a phylogenetic tree of a broad selection of jawed vertebrates [56].

same level. The second one refers to the phylogenies that include a molecular clock, that is, the branch length and their weights, if they are displayed, symbolize the number of *time units* that have elapsed between events (nodes). There are two major problems in this depiction: **i)** this representation assumes a *constant* rate of molecular changes across branches; and, **ii)** we have to relate at least one moment in the tree with a date to be able to give a meaning to those time units.

The way the tree topology is inferred and the data is settled relies on the methodology used, as we have claimed before. Nonetheless, the accuracy of these methods has a lot to do with their capacity to reproduce the evolution process that has affected the input data. An *evolution model* is a mathematical model that establishes, through different number of parameters and variables, the substitution rates of nucleotides or amino acids. Thus, multitude of evolution models have been proposed following different criteria: **i)** target sequences, i.e. DNA, RNA or proteins; **ii)** biological target, i.e. general purpose or specific kind of sequences, like the hmtDNA; and, **iii)** complexity, i.e. number of variables to calculate for the specific input dataset. As we will discuss in the next section, it is sometimes troublesome to know beforehand the best evolution model for our set of sequences. We will present the most applied alternatives to cope with this problem.

Although they are out of the scope of this thesis, we wanted at least to mention phylogenetic networks as alternative data structures for modeling phylogenies. They are interesting due to their associated complexity. The tree structure is very restricting for something as complex as evolution. The *phylogenetic networks* have been developed to extend the toolset of evolutionary events that can be modeled [57–59]. For instance, they provide a better image for recombinations: two or more mutations that occurred separately and they are merged again, making a cycle in the network.

4.2 Related work

In this section, we cover the methodologies developed to produce phylogenetic trees and supertrees from a preprocessed input dataset. We will also present the most well-known software tools developed for such purpose. Furthermore, at the end of this section, we will detail how we can measure and compare phylogenetic trees to determine the most accurate and reliable one for a given input dataset.

4.2.1 Phylogenetic inference

Next, we are going to detail the minimum evolution approach, where the number of differences between sequences is the metric used to determine the phylogeny; and the evolution model approach, where the algorithms rely on evolution models for the phylogenetic inference. There exist alternative inference methods that do not require the sequences to be aligned [60], but we will not cover them in this dissertation.

As we have claimed before, the way these problems are solved and the accuracy of the resultant tree will entirely depend on the methods and parameters selected for each input dataset [61–63]. However, we cannot overlook the $\mathbb{NP}$-*complete* nature of the problem of inferring the best phylogenetic tree [64], where the number of possible unrooted binary tree topologies is $(2n-5)!/\left((n-3)!\,2^{(n-3)}\right)$ [65], for n biological sequences. For instance, we could build more than 2 million different topologies from a set of only 10 sequences.

Minimum evolution methods

There are two main methodologies developed under the minimum evolution theory: the neighbor-joining method [66], and the maximum parsimony method [67, 68]. The NJ (***n**eighbor-**j**oining*) method generates a distance matrix for each pair of input sequences and uses a minimum-value search algorithm to define the inner nodes of the topology. The details of the method are described with pseudo-code in Algorithm 1.

The MP (***m**aximum **p**arsimony*) is an optimality criterion based on the minimization of the total number of mutations across the phylogenetic tree, that is, the shortest possible tree that explains the data is considered the best under this criterion. Nevertheless, the topology of the phylogeny is not inferred from the data: we need to apply another algorithm or methodology to accomplish this part of the problem. Given the enormous state-space of topologies we have presented before for just a few biological sequences, and

Algorithm 1: Neighbor-joining

```
   Data: $S$ (dataset)
   Result: $T$ (phylogeny)
 1 Create $T$ as a starlike tree from $S$;
 2 Define $M$ as a triangular distance matrix from each pair of strings of $S$;
 3 while $(rows(M) \geq 4)$ do
 4     Get $i, j \in M$, $i \neq j$ such that $M_{ij}$ is the lowest value;
 5     Join $T$ nodes $N_i$ and $N_j$ with new node $N_{ij}$ in between with their
         parent;
 6     Merge $N_i$ and $N_j$ in $M$ and recalculate it;
 7 end
```

Algorithm 2: Maximum parsimony

```
    Data: $S$ (dataset)
    Result: $T$ (phylogeny)
 1  For each sequence of $S$, create a leaf in $T$;
 2  while $(|S| \geq 2)$ do
 3      Get sequences $X_i, X_j$ with the fewest differences, $\forall i, j \in \{1, .., |S|\}$,
          $i \neq j$;
 4      if $\left(X_i \wedge X_j \notin T\right)$ then
 5          Join leaves $X_i$ and $X_j$ with a new inner node $N_{ij}$;
 6      else if $\left(X_i \oplus X_j \notin T\right)$ then
 7          Join the node of $T$ and the new leaf with a new inner node $N_{ij}$;
 8      else
 9          Join the two nodes of $T$ with a new inner node $N_{ij}$;
10      end
11      Replace $X_i$ and $X_j$ in $S$ for the consensus string of $N_{ij}$;
12  end
```

taking into account that a common input size is usually between 100 and 1000 sequences, this problem has been solved with heuristics. However, to get a better idea of how MP works, Algorithm 2 roughly displays the proposal made by Dr. Fitch [68] to solve both problems applying a dynamic programming approach. As we have claimed in Chapter 1, the MP problem has been demonstrated to be $\mathbb{NP}$-*hard* [10].

It is difficult to find recent software tools that implement only one of the aforementioned algorithms, like BioNJ [69]. Most of them are suites, gathering diverse alternatives for NJ and MP algorithms, altogether with other methods that we will introduce in the next section. An example of three widespread software suites that include at least one implementation for NJ and MP are T-REX [70], PAUP* [71] and MEGA [72, 73].

Methods based on evolution models

The lack of accuracy and other statistical downsides of MP and some distance-based methods led to the development of new algorithms founded on more complex evolution models. The first and most used one is ML (***maximum likelihood***) [74], which is an optimality criterion. Initially, it calculates the probability of the sequences' distribution throughout the given topology, followed by a maximization problem over all possible evolutionary trees. This method also assumes evolution independence among sites for each biological sequence. Algorithm 3 summarizes the ML method for a binary tree T with root s_0, where all the input sequences have been assigned to leaves of T. The length of any sequence of the input dataset is m. For a sequence s_k, we will denote with s_k^i the character at the i-th position of s_k. L will represent the likelihood of T, whilst L_{s_k} will represent the likelihood of the subtree of s_k from T. For a leaf s_k, $L_{s_k^i}$ will be equal to 1 when s_k^i equals the character we are checking, and 0 otherwise. The evolution model is mathematically decomposed in a prior probability vector, i.e. the probability of finding each character in a position of the sequences, and a mutation probability matrix. The goal is to maximize the likelihood score of T (L).

The ML problem has been demonstrated to be $\mathbb{NP}$-*hard* [75]. Besides, as it happened with MP, the topology must be calculated beforehand. The majority of the software tools implemented using ML apply heuristics to construct

Algorithm 3: Maximum likelihood

```
Data: $T$ (binary tree), $\pi$ (prior probability vector), $P$ (mutation
      probability matrix)
Result: $L$ (likelihood of $T$)
1 for $p \in \{1, \ldots, m\}$ do
2     for postorder tree traversal of $T$ do
3         Get $s_k$, parent node of $s_i$ and $s_j$ at $T$;
4         $L_{s_k^p} = \left(\sum_{s_i^p} P_{s_k^p s_i^p} L_{s_i^p}\right) \left(\sum_{s_j^p} P_{s_k^p s_j^p} L_{s_j^p}\right)$;
5     end
6     $L^p = \sum_{s_0^p} \pi_{s_0^p} L_{s_0^p}$;
7 end
8 $L = \sum_p L^p$;
```

the tree. Early efficient implementations for ML, like fastDNAml [76], have been replaced in most studies of the last decade by three ML software tools: PhyML [77, 78], RAxML [79, 80] and FastTree [81, 82]. The first one uses a fast distance-based algorithm to generate an initial topology, and after that, it implements a hill-climbing technique to adjust and modify the branches and their lengths to optimize the likelihood of the phylogeny. RAxML and FastTree use the same approximation to solve the problem: they generate k random topologies[1], afterwards they keep about the best 10% of them, in terms of a swift computed score, and finally, the algorithm computes the consensus tree to obtain the resultant topology. We will discuss the consensus tree and phylogenetic score concepts in the following sections. NJML [83, 84] offers another approach where the NJ topology building process is combined with the ML inference process. As it happened with MP and NJ, PAUP* and MEGA pose as two widely used examples of software suites intended for the phylogeny production that incorporate some of the aforementioned software tools.

[1]Although they choose huge values for k, it will always be small in comparison to the total number of topologies available in the state-space.

The model selection process is a crucial part of the configuration stage of ML. It is important to highlight that using the same model for all the possible input datasets may not produce an accurate phylogeny. On the other hand, testing every existing evolution model for each input dataset can be highly intensive in time cost. Therefore, choosing the best evolution model for the kind of data we are going to work with is one of the most important steps on the way to estimate a reliable phylogenetic tree. There are many articles about evolution models (e.g. JC69 [85], HKY [86] or GTR [87] models of nucleotides, and JTT [88] or WAG [89] models of amino acids), but it is easy to find many researches based on just one evolution model because it worked well on previous studies and experiments. For instance, FastTree offers only two nucleotide and three amino acid evolution models, whilst PhyML and RAxML accept almost any evolution model as an input parameter. However, FastTree has been proven to be faster and more accurate than RAxML, especially when dealing with large inputs [90]. There have been published specific software tools to determine, under different criteria, which is the best evolution model for each particular set of sequences in a reasonable time cost, e.g. ModelTest [91], or its newer version jModelTest [92], for nucleotide evolution models, and ProtTest [93, 94] for amino acid evolution models.

The second methodology proposed to infer phylogenetic trees using the evolution model approach is the BI (***b**ayesian **i**nference*) [95]. It states that the evolution model can be extracted from the topology and the distribution of the input dataset. The BI has become popular due to advances in computing speeds and the integration of MCMC (***M**arkov **c**hain **M**onte **C**arlo*) algorithms. It is based on a likelihood function to calculate the posterior probability of trees, that is, the probability of the tree to be correct, using an evolution model in the process. The BI is the most complex method of the four presented. The pseudo-algorithm presented in Algorithm 4 manifests the core idea of exploring the state-space until *enough* topologies have been tested to return the best phylogenetic tree.

Algorithm 4: Bayesian inference

```
   Data: $S$ (dataset)
   Result: $T$ (phylogeny)
 1 Build a phylogeny $T_i$ for $S$;
 2 for an enough number of iterations to reach an equilibrium point do
 3     Propose a new phylogeny $T_j$ for $S$, different from $T_i$;
 4     Determine the likelihood of $T_i$ and $T_j$;
 5     $T_i \leftarrow$ Apply Bayesian inference to get the best phylogeny;
 6 end
 7 $T = T_i$;
```

MrBayes [96, 97] is one of the most used software tools as to the BI method. Despite being a fast methodology, the BI methods have been proven to have an impractical time cost for large datasets [98].

Statistical robustness

At this point, it is necessary to acknowledge that the majority of the methods mentioned above (excluding BI) do not guarantee statistical robustness and, therefore, an extension of the phylogenetic analysis is required. In biology, this problem is usually covered by a bootstrapping [99] of the input sequences. The confidence intervals in the estimation process of phylogenies is achieved generating new datasets from the original one, wherein statistical character shuffling has been applied differently in each of them. Once the phylogenies have been estimated, a majority-rule consensus process must be applied to obtain the final phylogenetic tree.

Some of the tools we have mentioned in the previous section incorporate a bootstrapping option, e.g. FastTree or RAxML. For the others, an external software tool is required. The most extended one is Seqboot, from PHYLIP [100].

4.2.2 Phylogenetic assembly

There are just a few situations where the inference process would be the last step to obtain the final phylogeny from the selected biological sequences. In most cases, the bootstrapping is part of our phylogenetic inference workflow. Additionally, it is a common practice to divide the input dataset into smaller subsets for biological or performance purposes. The result in both scenarios is the same: we end up with more than one phylogenetic tree for the single input dataset we fetched. Therefore, we need to assemble these *partial* phylogenies to get the resultant evolutionary tree. Nevertheless, the way these phylogenies need to be processed is quite different. When a bootstrapping is performed, all these partial phylogenetic trees have the same leaves, thus, a *consensus tree* must be calculated. On the other hand, splicing the input dataset into smaller subsets, with or without overlapping, involves the construction of a *supertree*.

Consensus trees

As its name suggests, a consensus tree provides a convenient way to compile the agreement between two or more phylogenetic trees. It is built by combining subtrees that take place in at least a certain percentage of the input phylogenies. Despite being a plain concept, there have been published many different methodologies to cope with this problem. The most extended and used ones are the family of consensus tree methods called the M_l *methods* [101]. These include the strict consensus, the majority rule consensus and the greedy consensus trees. All these methods establish a threshold fraction l that has to be surpassed by a subtree in order to be included in the consensus tree. The strict consensus tree method states $l = 100\%$, whilst the majority rule one relaxes this condition to $l = 50\%$, including only those subtrees present in the majority of the phylogenies. The greedy approximation lowers the value of l below the 50%. Thus, the subtrees of the consensus tree are first ordered according to the number of times they appear, and then the consensus tree is built progressively to include all those subtrees whose support is above the threshold and that are compatible with the phylogeny constructed so far.

Other methodologies involve a more complex preprocessing of the available information on the phylogenies in order to build the consensus tree. A Nelson consensus tree [102] is the largest set of mutually compatible subtrees within the same set. If there is more than one solution set, the Nelson consensus tree is the set of subtrees common to all those solutions. Another well-known example is the Adams consensus tree [103], which takes into account not only the relationship between leaves, but also their depth in the phylogeny.

There have been many published algorithms for the aforementioned methodologies, and they are still being implemented in recent software tools [104–106]. For instance, the PHYLIP software suite offers a consensus tree builder tool named Consense.

Supertrees

The *supertree problem* [107] involves the production of a single phylogenetic tree (supertree) from a set of phylogenies with an overlapping subset of sequences. Thus, the evolution represented in those phylogenies is similar, but it does not concern to the same biological sequences. One of the first most widely used supertree algorithms in phylogenetics was MRP (***M**atrix **R**epresentation using **P**arsimony*) [108, 109]. First, the algorithm encodes the input phylogenies into binary characters and, afterwards, it uses MP to construct the supertree based on the previously obtained data matrix. There are two main problems involved in this algorithm. The first one brings back the same discussions about the simplicity of the evolution model applied in the MP method. Thus, other algorithms based on MRP but using ML [110] and BI [111] have been proposed.

The second problem with MRP is the neglect of the desirable property of every algorithm to be computed in polynomial time. As we stated before, the

MP method has been proved to be $\mathbb{NP}$-*complete*, and so it is MRP [112]. Thus, Dr. Semple and Dr. Steel developed a polynomial-time algorithm to compute supertrees from rooted phylogenies, named MinCutSupertree [113]. It is based on the computation of the minimum-weight cut set of a weighted graph that represents the relationship between all the leaves of the input phylogenies. The weight of each edge states the number of phylogenetic trees for which the pair of leaves joined by that edge are in the same subtree, that is, there is a path between them without going through the root of the phylogeny.

Finally, there is a proposed meta-method to fasten several supertree construction algorithms that cannot cope with large input datasets, that is, when the number of leaves surpass the few thousands barrier: SuperFine [114]. This software tool can reduce significantly the time cost of methods such as MRP, without having a negative impact on the accuracy of the initial algorithm.

4.2.3 Choosing the best phylogeny

As we have discussed in previous sections of this chapter, there are several software tools to infer the phylogenetic tree of a given set of biological sequences. Furthermore, most of these tools offer different parameterizations. Thus, a reasonable question arises: how can one choose the best inference method and parameterization? In other words, we want to produce the best phylogentic tree for our input dataset. We have summarized the ML method in Section 4.2.1. This is the main configurable approach that includes a score computation during the inference process. The likelihood score is computed for each site but, since those values are extremely small numbers, it is convenient to sum their log likelihoods and report the score of the entire phylogeny as the log likelihood. Therefore, the best phylogenetic tree will be the one with a closer log likelihood score to 0. It is important to remark that as the complexity of the evolution model increases, it is expected to get an improvement in the final score. Nonetheless, this enhancement might be the result of an over-parameterization that simply fits the model to noise in the data. There are a few software tools, like MrBayes in its last version (v3.2) [97], that have included a ML score computation process to solve that problem. Of course, another metrics can be applied: the maximum parsimony, with the aforementioned drawbacks, and the Bayesian posterior probability, which has been demonstrated to be unreliable under certain conditions, like the complexity of the evolution model selected [115].

The assessment of the accuracy of new algorithms and methods is usually performed with synthetic biological sequences and phylogenies. In general terms, the whole data is constructed from a *random* topology for which a specific evolution model is selected to generate the biological sequences of its leaves. Thus, the *true phylogenetic tree* altogether with its evolution model is known a priori for the given set of biological sequences. The process is detailed in the supplementary material of [23]. Afterwards, every new tool can be compared not only in terms of performance and memory consumption, but also in terms of accuracy. In this case, besides the score function, we can use the comparison of the inferred phylogenies with the true one. The most extended and used metric is the Symmetric Difference of Robinson and Foulds [116]. It uses only the topologies of the trees to count the average between FP (***f****alse* ***p****ositive*) and FN (***f****alse* ***n****egative*) rates, that is, the proportion of internal branches appearing in the inferred tree that are not in the true phylogeny (FP) versus the proportion of internal branches in the true tree that are missing from the inferred phylogeny (FN). There are other metrics, like the Branch Score Distance of Kuhner-Felsenstein [61], which takes into account not only the topology, but also the branch lengths. There are several software tools intended to compare phylogenies, e.g. Treedist from PHYLIP [100] or PhyloNet [117].

4.2.4 Machine learning meets phylogenetics

Among all the approaches published that apply diverse machine learning techniques in the production of phylogenetic trees, we found a very efficient algorithm proposed by Dr. Ishteva [118] using fourth order tensors to learn the latent tree topology. This algorithm offers a different methodology involving the concept of evolution models that is able to avoid the heuristic approximation regarding the topology construction. We present this algorithm here and explore its implementability in Section 4.4.

The core of this proposal relies on another family of provably consistent algorithms: the quartet-based methods [119, 120]. They first resolve a set of relations for quadruples of biological sequences, called *quartets*, and subsequently, stitch them together to form the phylogenetic tree. Determining the quartets accurately is essential in these methods, as the assembling algorithm calls that procedure repeatedly. Recent publications [121] have proposed quartet algorithms focused only on the leading k singular values of the joint probability table, where k is the number of possible characters at each site of the sequences at the inner nodes of the tree. Despite the advantage these approaches offer allowing k to be different from the observed number of characters of the given sequences, k is still an input parameter of these methods. Thus, the machine learning algorithm that drew our interest has developed a new approximation *agnostic* to k, since in practice we rarely know this number.

The algorithm uses the proposed quartet-discovery subroutine and recovers the latent phylogeny applying a divide-and-conquer strategy. Another valuable property of this algorithm is its computational complexity: $\mathrm{O}(n \log n)$, for n input biological sequences. This makes the algorithm suitable even for very large problems.

4.3 MEvoLib's phylogenetic interfaces

In this section, we cover two interfaces we have included in MEvoLib to provide a fully configurable and up-to-date solution for using phylogenetic production software tools by means of Python scripts.

4.3.1 *MEvoLib.Inference*

The *Inference* interface handles the phylogenetic inference process with widely used software tools. As with *Align*, this interface has a single parameterization for every inference tool included in MEvoLib. The user must provide the desired phylogenetic estimation tool, the input sequence file and its format, the list of arguments of the tool and the number of bootstraps to generate during the inference process. This last parameter is only available for those tools that include this functionality. The resultant phylogeny will be returned in a *Tree* object, from Biopython, together with its log-likelihood score. The list of parameters can be extended to include the output tree file and its format, where the phylogenetic tree will be saved. *Inference* also includes the automatic format conversion functionality, which facilitates the interaction of MEvoLib with other methods and software tools.

This interface implements a dictionary of common configurations for each included tool too. Apart from the *default* configuration, which is used when no list of arguments is presented, all the keywords follow the same nomenclature: [*evolution model*]+[*approximation method*]. It has been designed to resemble the model applied with the predefined configuration, improving both the code readability and the keyword memorization. New configurations included by the user may ignore this nomenclature.

The current version of MEvoLib supports the latest version of two of the aforementioned phylogenetic inference tools: FastTree and RAxML.

4.3.2 *MEvoLib.PhyloAssemble*

Although the *PhyloAssemble* interface is intended for consensus tree and supertree estimation processes, in the current version of MEvoLib only the consensus tree estimation process has been implemented. To get the consensus tree from a set of phylogenetic trees, the interface requires the consensus tree tool, the input tree file, the file format, and the list of arguments of the tool.

The resultant consensus tree will be returned in a *Tree* object, from Biopython. It can also be saved as a tree file including in the parameter list its file path and format. *PhyloAssemble* also includes the automatic format conversion functionality.

This first version of MEvoLib incorporates a modified version of the Consense method for the consensus tree estimation process. This software covers the strict consensus and the majority rule consensus tree methodologies, and an algorithm similar to the Nelson consensus tree method. Its source code has been changed to allow two new arguments: the input and output file paths. The reason for such a decision is that the default behavior of Consense is to read a file named *infile* in the current working directory and to write the consensus tree in a file named *outfile* in the same directory; this may lead to execution problems in infrastructures like clusters. For instance, the working directory in the node running the job might not be writable by the user, raising an error at runtime.

4.4 Learning the latent DNA phylogeny

As we were not able to obtain a working implementation of the algorithm mentioned in Section 4.2.4, we proceeded to develop a new implementation in collaboration with Dr. Balle and Dr. Requeno. Dr. Balle helped us to fully comprehend the routines of the algorithm related with machine learning techniques, whilst Dr. Requeno assissted us in the implementation of the method in C++. As we have aforementioned, the algorithm is based on the correct composition of the quartet relationships. It uses a 4th order tensor approximation to build every possible quartet for four given sequences. Afterwars, the algorithm applies a SVD (***s**ingular **v**ector **d**ecomposition*) to choose the best quartet. We have taken advantage of the Eigen library [122] to implement this subroutine.

Once the algorithm was implemented and Dr. Balle verified it, we wanted to compare its accuracy with other approximations. We chose the synthetic datasets published by the research group of Dr. Warnow in 2009 [23] as input. These sequences have been created following a specific topology and evolution model. Therefore, the “real” phylogenetic tree is known beforehand and we can measure the algorithm’s accuracy in terms of its result’s divergence from the expected phylogeny. We also used the time cost results of RAxML and FastTree published in the same paper as a means to picture the efficiency of our implementation in the current state-of-the-art. Whilst we got promising results in terms of efficiency, being comparable with those of RAxML, the resultant phylogenetic trees were highly inaccurate in every case. After a deep study on the plausible causes, we realized that the accurate results showed in the paper of the algorithm we implemented are based on the prior knowledge of the probability distribution of the input sequences. None of the proposed subroutines infers this input distribution, and in real case scenarios, this parameter is unknown. Besides, computing it separately would make the resulting method much slower. We consider this omission a flaw in the proposed algorithm.

4.5 Conclusions

In this chapter, we have presented two interfaces of MEvoLib designed to cope with the production of phylogenetic trees based on preprocessed biological sequences. *MEvoLib.Inference* has been designed to provide a fully compatible and configurable interface to facilitate the work with phylogenetic inference software tools. On the other hand, *MEvoLib.PhyloAssemble* achieves the same objectives for phylogenetic assembling software tools, i.e. consensus tree and supertree building methods. Both interfaces also include the format conversion functionality, very useful for the integration of different software tools in a phylogenetic inference workflow. Moreover, we have covered a different approximation to solve the phylogenetic inference problem. The application of the machine learning technique aforementioned to discover the latent tree topology from a set of biological sequences led to a dead end due to a plausible error in the published algorithm.

As future work, we plan to extend the software tools available for both MEvoLib interfaces. For instance, we aim to incorporate PhyML for ML, a bootstrapping method like Seqboot of PHYLIP to apply when the inference method cannot generate bootstraps by itself, and SuperFine as an example of a supertree construction tool. Other methods for alignment-free phylogeny production will also be considered.

I love fools' experiments. I am always making them.

Charles Robert Darwin

5 Phylogenetic inference systems

In the previous three chapters of this dissertation, we have presented all the stages usually involved in phylogenetic inference studies, from the fetching of the dataset to the estimation of the phylogeny. An implementation of a complete phylogenetic inference workflow integrating standalone software tools and the methods already presented can be harsh, especially for novel researchers without adequate programming skills. Moreover, multitude of published works rely on similar schemes of the aforementioned procedure but each research group usually implements them from scratch again for their specific case of study. All those expenses could be saved if those blueprints were implemented with some tune-up adjustments and applicability for different scenarios. That has been the foundation for the creation of phylogenetic inference systems for both specific and general purposes. Additionally, the production of phylogenies, principally those involving large datasets, take several hours or even days of computational work to be completed. On the other hand, several tens or hundreds of new biological sequences are uploaded every month, driving the phylogenies to an outdated state in just a few months. Thus, it is not economically neither temporally affordable to produce phylogenies at the information's update pace.

In the initial section of this chapter, we present two cases of how the phylogenetic inference workflow can be performed without the usage of phylogenetic inference systems. The second section gathers a prospection in phylogenetic inference systems, including details about the computer architectures in which they rely on. We focus our attention on the most relevant systems published and, specifically, on how the resultant phylogenetic trees could be updated to incorporate new information. In the following two sections, we present our contributions to this topic: a phylogenetic inference framework that is automatic and fully customizable, and a new software tool to update phylogenies even in the large case scenario. These contributions have been published in the conference papers [123–125] and [126], respectively. Finally, the last section gathers the conclusions for this chapter.

5.1 Step-by-step workaround

At this point, we are able to infer the phylogenetic tree we desire from scratch selecting the adequate software tools from all the alternatives introduced. For such purpose, we need to link manually our phylogenetic inference procedure or use a phylogenetic software suite like MEGA [72, 73] or PAUP* [71, 127]. Following, we provide two different examples on how it could be done. The first one develops a new Python script for the study of the evolutionary relationship of two hmtDNA genes. It will use only MEvoLib [28] and Biopython [13] to achieve its objective. The second example examines the available amino acid evolution models before inferring the phylogenies for the protein-encoded genes of hmtDNA. This last case is based on the work performed by Enrique Miguel during his undergraduate project in 2012 [128]. In this occasion, command-line calls and independent scripts handle all the methods.

Studying *MT-ATP6* and *MT-ATP8* genes with *MEvoLib*

We are going to study the molecular evolution of two hmtDNA genes: *MT-ATP6* and *MT-ATP8*. The example will illustrate the potential and limitations of MEvoLib following the workflow presented in Figure 5.1. The question we pose is if they have evolved similarly based on the following information: **i)** both genes encode proteins that belong to the same subunit of the ATP synthase enzyme; and, **ii)** these two genes are located in the same strand and they have an overlapping of 42 nucleotides.

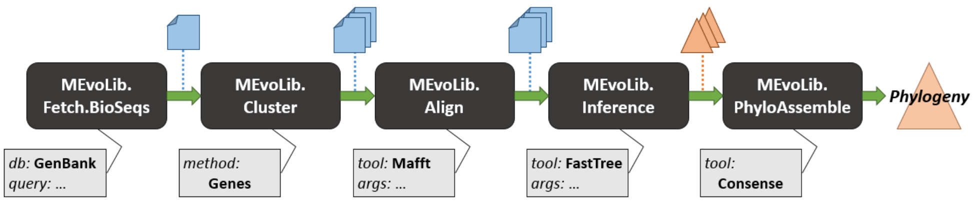


Figure 5.1: Phylogenetic inference workflow implemented with MEvoLib.

First, we are going to download all the hmtDNA sequences from GenBank that have any information related with the *MT-ATP6* or *MT-ATP8* genes. The basic query to achieve this purpose would be: *"homo sapiens"[porgn] AND mitochondrion[Filter] NOT mRNA[Filter] AND (atp6 OR atp8)*. However, based on the information we have gotten in the study of the *Genes* method we have presented in Section 3.3.3, these genes are referred with diverse nomenclatures. Therefore, our fetching source code is:

```
from MEvoLib.Fetch.BioSeqs import BioSeqs
seq_db = BioSeqs.from_entrez(
    email="eg@test.com",
    entrez_db="nuccore",
    query='"homo sapiens"[porgn] AND
       mitochondrion[Filter] NOT mRNA[Filter] AND
       (atp6 OR atpase6 OR "atpase 6" OR "atp
       synthase 6" OR "atpase subunit 6" OR "atp
       synthetase subunit 6" OR "atp synthase f0
       subunit 6" OR "atp synthase fo subunit 6" OR
       atp8 OR atpase8 OR "atpase 8" OR "atp
       synthase 8" OR "atpase subunit 8" OR "atp
       synthetase subunit 8" OR "atp synthase f0
       subunit 8" OR "atp synthase fo subunit 8")')
seq_db.write("hmtDNA_ATPs_all.gb")
```

We were able to download 32548 sequences with the first query, whilst the second one fetched 32626 sequences (on 07/Jul/2016). In the downloaded dataset are two hmtDNA sequences that have errors in their biological information, puzzling the genes' extraction. Hence, we are going to remove them from the downloaded dataset:

```
del seq_db.data["DQ862537.1"]
del seq_db.data["FR695060.1"]
seq_db.write("hmtDNA_ATPs_filtered.gb")
```

Next, we need to extract the genes we are interested in from the filtered dataset. For this example, we are going to focus exclusively on the *CDS* feature. In a complete study, we would also be interested in the *gene* feature to include those sequences that might have only tagged the genes under this feature. The following source code presents the gene extraction applying the *Genes* method:

```
from MEvoLib import Cluster
gene_dict = Cluster.get_subsets("genes",
   "hmtDNA_ATPs_filtered.gb", "gb", ["CDS"])
```

Afterwards, we extract those elements in the dictionary that refer to the *MT-ATP6* and *MT-ATP8* genes and we save the corresponding sequence fragments in two separated FASTA files. The software tool we are going to use to infer the phylogenetic trees expects an alignment as input. Thus, the following code implements the MSA procedure for the *MT-ATP6* gene using MAFFT [29] in its default configuration (`--auto`):

```
from MEvoLib import Align
Align.get_alignment("mafft", "atp6.fasta", "fasta",
   args="default", outfile="atp6.aln",
   outfile_format="fasta")
```

The next stage in our workflow is the phylogenetic inference. We are going to use the *Inference* interface from MEvoLib, selecting the FastTree tool [81] in its default configuration (*GTR+CAT evolution model*). The resultant *MT-ATP6* gene tree and its log-likelihood score will be returned, and the tree will also be saved in a NEWICK file:

```
from MEvoLib import Inference
tree, score = Inference.get_phylogeny("fasttree",
   "atp6.aln", "fasta", args="default",
   outfile="atp6.newick", outfile_format="newick")
```

The inference workflow for this research would be completed with these last results. To extend our study to include the last stage of the workflow shown in Figure 5.1, we could be interested in analyzing the consensus tree composed from the *MT-ATP6* phylogenies of different species, using the Consense method from PHYLIP [100] in its default configuration (*majority rule consensus*):

```
from MEvoLib import PhyloAssemble
PhyloAssemble.get_consensus_tree(
    "consense", "atp6_species.newick",
    "newick", args="default",
    outfile="atp6_species.cons",
    outfile_format="newick")
```

Human mtDNA phylogenies and their protein counterparts

This study starts with the fetching of biological sequences, like the previous example. In this case, we downloaded 14915 hmtDNA sequences form GenBank which length was in between 16000 and 17000 nucleotides. Those were all the sequences available when this study was performed, back in 2012. Afterwards, the sequences were pairwise aligned with the rCRS to determine the location of the protein-encoded genes of each sequence. Enrique Miguel implemented a script to perform the alignment automatically using MUSCLE [46] v3.8.31 in its default configuration. The script extracts sequentially each downloaded sequence and aligns it with the rCRS. Once the alignment is completed, the script removes those sites of both sequences corresponding to gaps introduced in the rCRS during the alignment process. Thus, the genes' starting sites will be the same as those in the rCRS. The script took a week of work in a regular desktop computer to complete the task. Since we wanted to get the protein encoded in each hmtDNA gene, we were concerned about the possibility of finding gaps in the genes introduced by the alignment tool. This would entail a problem in the translation stage, where we replace each codon in the RNA string by its corresponding amino acid, leaving at least one codon without the chance of being correctly translated. An analysis of the aligned sequences returned 6 sequences that matched the presented issue, and they were removed from our dataset. Afterwards, we separated each protein-encoded gene from their sequence and we finished this stage with 13 gene files.

A second script was developed to perform the transcription and translation processes for each sequence. The former just replaces the T nucleotide by its corresponding RNA value (U) except for the gene ND6, which requires to be inverted beforehand. This inversion involves not only reading of the sequence from the end towards the beginning, but also replacing each nucleotide by its complementary: A by T, C by G, and vice versa. The latter translated each codon by its corresponding amino acid. If we found a codon with an N

character, which represents that there is a nucleotide but we ignore which one is, it was translated by X. It has the same function in proteins as *N* in DNA. If the gene sequence was incomplete, it was filled with X values in order to have all the substrings with the same length.

Finally, a model selection process was performed in order to estimate the best phylogeny for each protein. For this task we chose ProtTest [93] v3.0. Among the 120 amino acid evolution models tested in that version, and the 4 different evaluation methodologies included, approximately an 80% of the cases returned the model *MtMam* [129] as the best description, and another 10% selected *mtREV* [130] as the best one. These results agree with diverse studies published about the three most suitable evolution models for mitochondrial proteins in vertebrates [131]: *mtREV*, *MtMam* and *MtArt* [132]. Thus, we used the *MtMam* evolution model with RAxML [79] v7.0.3 to infer the phylogenies of each protein dataset.

5.2 Related work

As we have manifested with the two previous examples, researchers have to perform an adequate state-of-the-art analysis and review several software tools in order to achieve accurate results for their phylogenetic studies. Moreover, basic programming skills are advisable to arrange automatically the input and output data of each procedure. Whilst the former is always desirable for a good usage of the methodologies, the latter should be spared in most cases. Putting aside the learning time cost, the resemblance of the scripts developed for many evolution studies represent an expense that could be easily avoided. For instance, it is almost a routine to develop scripts to convert between file formats to adjust the output of one software tool to the input of the next one.

Besides, using large datasets as input has disclosed overflow problems when stressing conventional methods and tools. It has been shown that some of them turn to be inaccurate when moving from small datasets to big ones [90]. Parallel execution is one of the most suitable techniques that can be applied to reduce running time. Efforts in this direction have been focused on fine-grain parallelization of standard algorithms and application of algorithmic engineering techniques to improve implementations [76, 79]. Some of these undertakings have harvested remarkably good performance measures, but nonetheless, typical algorithms were not designed bearing in mind the concurrency. Thus, the number of independent tasks at a given moment is limited, as it is their individual load. Moreover, the computation of large and accurate phylogenies remains as one of the most challenging fields in the area [133] because the size of the input data and the complexity of the model pose great statistical and computational challenges [134].

The phylogenetic inference procedure we have been thoroughly exploring in the past chapters states a prevalent scheme in most molecular evolution researches. The two cases presented in the previous section portray an excellent example. Workflows have become a widely spread abstraction for sharing complex scientific methods. They capture experimental methods designed by scientists to test a given hypothesis and translate them into a pool of coordinated computational tasks in an automated execution environment [135]. Hence, workflows have been key to design phylogenetic inference systems. Working with evolution models using both intensive data and computing resources has generated a growing interest in building global resources for computing, sharing knowledge about their workflows of tasks as well as big databases of biological data. This has led to exploit the capabilities of both specific hardware platforms, like FPGAs (***F****ield* ***P****rogrammable* ***G****ate* ***A****rray*), GPUs (***G****raphics* ***P****rocessor* ***U****nit*) and CBEs (***C****ell* ***B****roadband* ***E****ngine*), and HPC (***H****igh* ***P****erformance* ***C****omputing*), HTC (***H****igh* ***T****hroughput* ***C****omputing*) or Cloud facilities available [136–138]. Furthermore, we cannot disregard the unaffordable time cost that emerges when performing exhaustive phylogenetic inference studies, even when we take advantage of highly parallelizable architectures.

Next, we will perform a further exploration on the computing platforms most phylogenetic inference systems are based. After that, we will present four of the most well-known phylogenetic inference systems published. Finally, we will cover the problem of the elevated time cost of large-scale phylogenetic studies.

5.2.1 Computing platforms

The following review about computing platforms, paying especial attention to those useful for large phylogeny inference tasks, has been organized according to three main levels of abstraction: cloud computing approaches, ad-hoc hardware devices and HPC/HTC facilities.

Cloud computing infrastructure is becoming widely used because Web Services offer the opportunity to gain flexible, on-demand, access to remote storage, processing power and other hardware facilities. This way the majority of computational processing occurs remotely in external service providers [139]. In the bioinformatics community, Taverna [140, 141], Galaxy [142], Tavaxy [143] and bioKepler [144] are some of the most successful suits for deploying workflows in the cloud. The growth in this field has impelled the availability of workflow repositories which provide useful resources for developing new analysis methods [136]. Nevertheless, due to the prominent advantages of workflow approaches based on cloud computing, it cannot be neglected that there still exist big challenges in standardization of formal workflow specifications as well as in managing the heterogeneity and ever-changing nature of distributed services [141, 143, 145].

Hardware accelerators based on FPGAs, GPUs and CBEs provide meaningful support for exploiting the specific granularity levels of parallelism, which are inherent to widespread algorithms, used in repetitive tasks in molecular biology. These platforms can provide both speed-up and energy efficiency improvements when compared to conventional computing architectures based on general purpose processors. FPGA-, GPU-, or CBE-based systems can achieve better performance than those based on CPUs on certain workload. In the context of phylogenetic inference, it can be noticed that well-known costly algorithms such as ML have been a spotlight for their implementation in all the aforementioned computing architectures [146–148]. Nowadays, these devices have become object of interest because they can be much more easily prototyped than ad-hoc multicore processors integrating large number of processing elements on a single chip [149]. Nevertheless, as coprocessors based on novel CPU approaches such as the Intel MIC (***M**any **I**ntegrated **C**ore*) architecture are becoming available, standard x86 development toolchains (*icc*, *pthreads*, *OpenMP*) can be used straight away to implement ML [150].

The wide spread use of HPC and HTC facilities have provided an easy access to biologists to a quite simple execution of parallel phylogenetic inference tasks. From the technical point of view, one of the most challenging issues in the field is how to scale widely used tools according to the growth of input datasets [151]. In this context, load balancing for both performance and power efficiency remain as some of the most challenging issues [152], together with improving concurrent data access to datasets with exponential growth in terms of complexity and volume [153]. For the particular case of core phylogenetic inference algorithms such as ML, there exists a big interest in developing highly parallel implementations capable of improving resource exploitation in HPC facilities [154]. Finally, concerning cloud computing platforms, it can be noticed that one of the most challenging issues is to develop meaningful interoperability between HPC and ABDS (***A**pache **B**ig **D**ata **S**tack*) architectures [155] as well as their integration with cloud computing approaches [137].

5.2.2 Prearranged solutions

In the last years, diverse phylogenetic inference systems have been published to estimate fast and accurate phylogenetic trees. SATé-I [23], SATé-II [24] and DACTAL [25] are good examples of general-purpose systems, performing efficiently even with large datasets. The workflow of these three systems relies on the circular dependency of the phylogenetic inference methodologies that require a MSA as input. As it is shown in Figure 5.2, we cannot expect to infer a reliable phylogenetic tree if the input MSA is inaccurate. Moreover, a better incremental MSA based on evolution relationships cannot be computed with a vague phylogeny. Thus, both versions of SATé and DACTAL have an iterative-refinement design where, once an initial MSA and phylogenetic tree are computed, the $(i-1)$-th phylogeny is used to improve the accuracy of the i-th MSA, and that new result is used to infer a more precise phylogenetic tree. The process finishes when the i-th phylogeny does not represent an enhancement to its previous one, or when a time limit is reached.

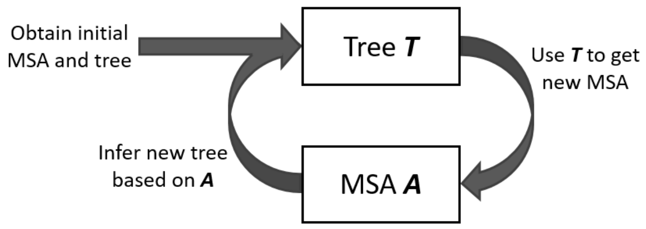

Obtain initial MSA and tree
Tree ***T***
Use ***T*** to get new MSA
Infer new tree based on ***A***
MSA ***A***

Figure 5.2: Existing accuracy-improvement feedback between phylogenetic inference and MSA processes.

The design foundation of SATé is a divide-and-conquer strategy. After the initial phylogeny is estimated with ML, the different non-overlapping subtrees of up to a maximum number of leaves are extracted. Then, the system computes the MSA of each subset of biological sequences. Following the tree topology backwards, the smaller MSAs are merged until the system obtains the full MSA. Finally, a new phylogenetic tree is estimated using again ML with the new MSA. The main differences in both versions of SATé are the methodology used for the subtree division and the MSA merging tool. Despite applying the same divide-and-conquer approach, the way DACTAL carries out the map and reduce procedures diverges. Whilst SATé estimates both the MSA and the phylogenetic tree for the given input dataset of biological sequences, DACTAL focuses solely on the inference task. First, the system computes a starting MSA and phylogenetic tree. Afterwards, the phylogeny is decomposed into overlapping subtrees with a prefixed limit in the number of leaves per subtree and the amount of overlapping leaves between subtrees. Next, a MSA and a phylogeny are computed for each subset. Finally, a supertree method, i.e. SuperFine [114], is used to merge the *subphylogenies*, estimating the final phylogenetic tree. This last result serves as input for the next iteration, as it happened with the previous systems.

ZARAMIT [26, 27] constitutes an excellent example of specific-purpose systems. It infers the phylogenetic tree from a set of complete hmtDNA sequences. The workflow is an acyclic graph composed by the main stages presented for a common phylogenetic inference process. ZARAMIT computes the MSA by a pairwise alignment of each hmtDNA sequence with the rCRS, following the same algorithm we introduced in the second example of the former section of this chapter. Taking advantage of the hmtDNA haplogroup knowledge, it arranges an MSA for each haplogroup. This step reduces the total computational cost of the inference stage. Next, a model selection process is applied to each subset to determine their best evolution model. Afterwards, the system infers each phylogeny with the resultant model. Finally, the hmtDNA haplogroup tree is used as template to join the phylogenies. Another relevant example is PhyloTree [41], which provides a detailed hmtDNA haplogroup phylogeny reconstruction almost yearly. The main issue about this system is its privacy: there is, to our knowledge, no system design or technical details available, apart from the lack of information about the source of the biological knowledge applied to reconstruct the hmtDNA phylogeny.

All these systems grant a fair install-and-launch platform based on fixed configurations used for each stage of the inference process according to a set of given parameters. This outlines a lack of flexibility for end-users because they can neither change the parameters/tools nor modify or adapt the system's workflow[1]. Furthermore, SATé and DACTAL disregard the available biological knowledge or special techniques for specific data. For instance, the use of evolution models that reflect the specific patterns of change observed in each dataset is crucial for obtaining realistic phylogenies, but the model selection process has been avoided in all of them.

5.2.3 Update phylogenies without rebuilding

The year gap in between updates of the hmtDNA haplogroup phylogeny of PhyloTree endorses the associated time and economic costs of inferring very large phylogenetic trees. In the meantime, hundreds or even thousands of new biological sequences are sequenced thanks to NGS technologies. Hence, new researches can be delayed for more than a year waiting for the next update of the phylogenies or, even worse, their hypotheses can be based on outdated information concluding with misguided results.

On the other hand, due to the fast availability of new biological sequences, most of them cannot undergo independent curation: the metadata are neither standardized nor homogeneous. Hence, the individual quality of a sequence, measured as its accuracy with respect to the original copy, is a priori unknown. Sequencing errors, that is, the set of sites where the value differs from its counterpart in the original sequence, may occur due to contamination —as for the rCRS [55]— but also because NGS techniques. They replicate very small segments of DNA and sort them together by local alignments, in which shorter segments are more susceptible to yield false positives. The error rates of current technologies, known in some cases [156], unknown in others, are far away from negligible. In addition, contamination is not a measurable factor in isolation.

Both problems can be solved using the evolutionary information for the detection of sequencing errors whilst we add the new sequences to the phylogeny within global updates. A phylogeny allows grouping each new

[1]Based on our experience, at least it is not easy to perform such a task.

sequence with its close relatives and measuring similarity between these and their ancestors. Representative mutations of each group are respected almost universally, and exceptions to this conservation are almost certainly due to errors in the sequencing process. Although it is possible to discover new subgroups and unusual variations, the probability of these facts will depend on the current state of the phylogenetic tree. The resultant phylogeny will not be as accurate as a newly inferred one, but it is the best alternative among the options.

There is, to our knowledge, no previous work on automatic sequence evaluation using evolutionary information. Nevertheless, there do exist some methods for the placement of sequences into a phylogeny. *pplacer* [157] is a software application designed for the phylogenetic placement of biological sequences. It uses some techniques like ML and BIC (***B****ayesian* ***I****nformation* ***C****riterion*) [158] to select the closest node of the reference tree to the sequence. Unfortunately, it works with a specific type of data called *metagenomics*, making it difficult to be applied to our target data. Another placement tool we have found is part of the software toolkit of the *Ribosomal Database Project* [159]. Its main drawback is that it only works with ribosomal RNA sequences, so all the processes applied are of specific purpose. Additionally, it is only available online.

5.3 PhyloFlow

PhyloFlow [123–125] is a flexible framework to design and customize phylogenetic inference workflows that can be deployed in workstations (standalone version) and clusters. There exists another version of the same system based on SaaS (***S****oftware* ***as a S****ervice*) made by Alvaro Recuenco and advised by Dr. de Miguel Casado and Dr. Fabra [160]. PhyloFlow is based on a previous phylogenetic inference system we published in 2011 [161, 162], which was able to perform the phylogenetic estimation of large datasets using model selection. PhyloFlow improves this previous system including the capability of building far more flexible and customizable systems, among other new features. Next, we introduce the design and implementation of PhyloFlow. After that, we show the study of the workload we have performed of every component of the system, in terms of execution time and memory. Finally, we present two practical applications of PhyloFlow.

5.3.1 Design and implementation

The design of PhyloFlow follows a *black box methodology* that provides the appropriate modularity to assess the requirements for the on-the-fly parameterization and tool-and-deployment flexibility addressed. Moreover, it has been developed to ease the addition of new tools: the existing ones not already included or those who will come out in the future. As shown in Figure 5.3[2], it has a two-part design: the configuration front-end and the phylogenetic analysis system itself. We refer from now on as phylogenetic analysis to the workflow involving the aforementioned phylogenetic inference procedures and the processes involved in the posterior analysis of the results. We will cover these analyses on the next chapters of this dissertation. Once the user defines a specific configuration for the workflow to be deployed, the resultant system can be launched. The system will handle automatically the interaction between the chosen stages, preventing the concerns of the user to this matter and providing a high robustness in the ad-hoc system built. The design also includes a *stage database* that works as a checkpoint to allow the system to recover from errors and shutdowns of the hardware system without the need of starting again from the beginning. Furthermore, those backups also provide a huge feedback of what has been done in each step, in case the user wants to reproduce any experiment or check them by other means.

[2]All the design images have been standardized using BPMN (***B**usiness **P**rocess **M**odel and **N**otation*), an expressive process modeling language that has already been used in the bioinformatics domain [163].

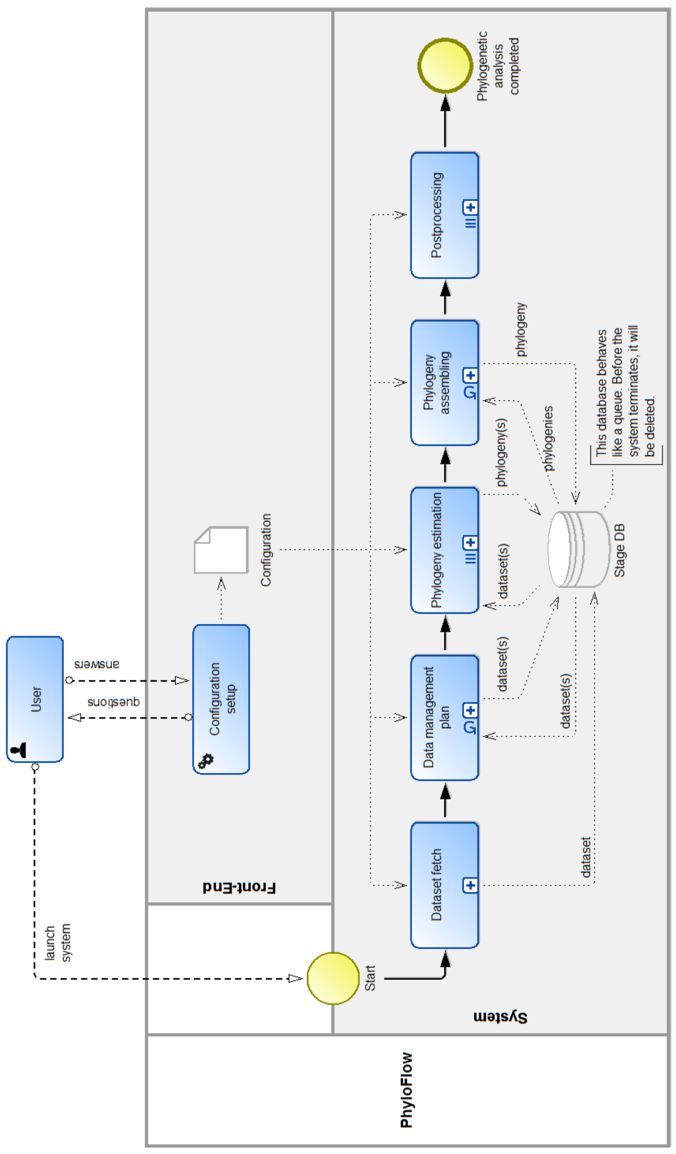


Figure 5.3: Design of PhyloFlow in BPMN. The Front-End handles the configuration of the phylogenetic inference system. The framework has five modules, each representing one of the aforementioned stages that are part of most phylogenetic analyses.

PhyloFlow has been implemented using HTCondor [164] and its meta-scheduler DAGMan [165]. This decision facilitated the transition from the design to the implementation. Additionally, the HTC of HTCondor grants the parallelization of massive amounts of processes in each phase, reducing considerably the time cost of each module. This technology, in combination with the data decomposition processes, allows handling very large datasets with high accuracy. This is based on the same concepts as DACTAL or SATé. The iterative refinement basis of the aforementioned systems has been implemented with a meta-script that controls the execution of the workflow. All the scripts implemented in PhyloFlow have been programmed in Python and they are compatible with both v2.7 and v3.3.

The Front-End aims to help in the translation of the user's desired experiment into a file that the processes involved can read and understand. This objective is achieved by a question/answer procedure involving the end-user and the configuration setup process, where the description of the tasks that relate tools, parameters and input-output files for each stage are defined. Besides, the interaction between stages can be modeled according to the requirements of the workflow scheduler for a given platform (e.g. standalone, cluster, grid, cloud,...) or even for mixed configurations. Once the configuration process has been completed, the system is ready to be deployed and executed.

The former stage of PhyloFlow consists in the automatic fetching of the input dataset of the configured system. As we have stated before, many studies in the field of phylogenetics are based on evaluating existing solutions for new datasets [166], or testing different parameters or methods using well-known datasets to perform a subsequent comparison with previous results [90]. Figure 5.4 shows the design of this stage, fetching the dataset(s) from local file(s) or from a biological database, like GenBank [17]. The backbone of this task during the implementation was the selection and incorporation of biological databases. Currently, the system allows to access GenBank through the *Entrez*

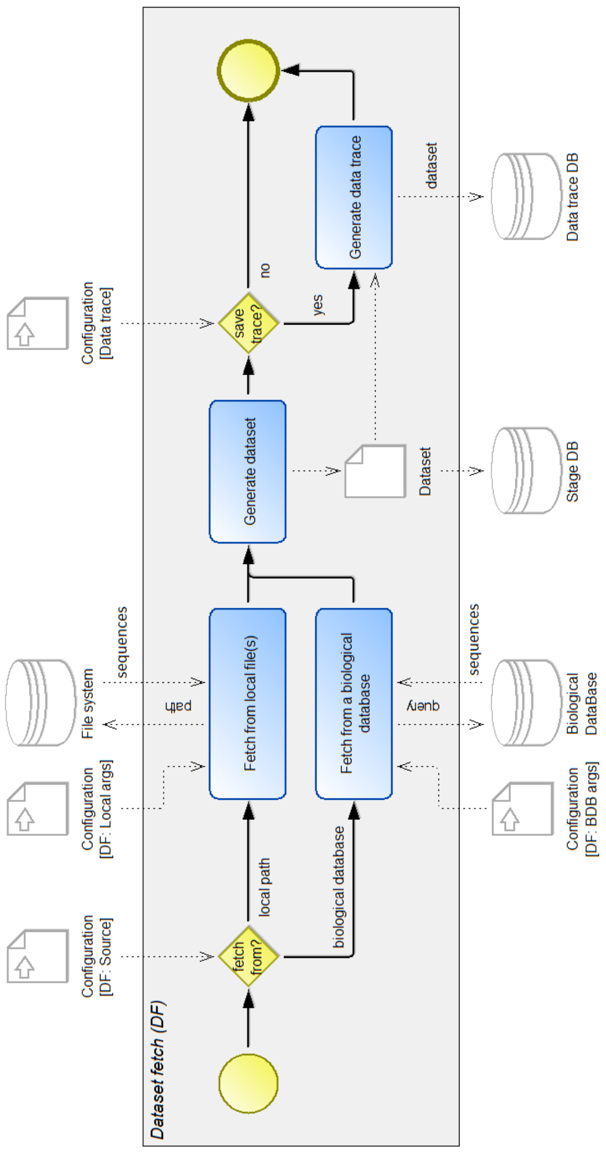
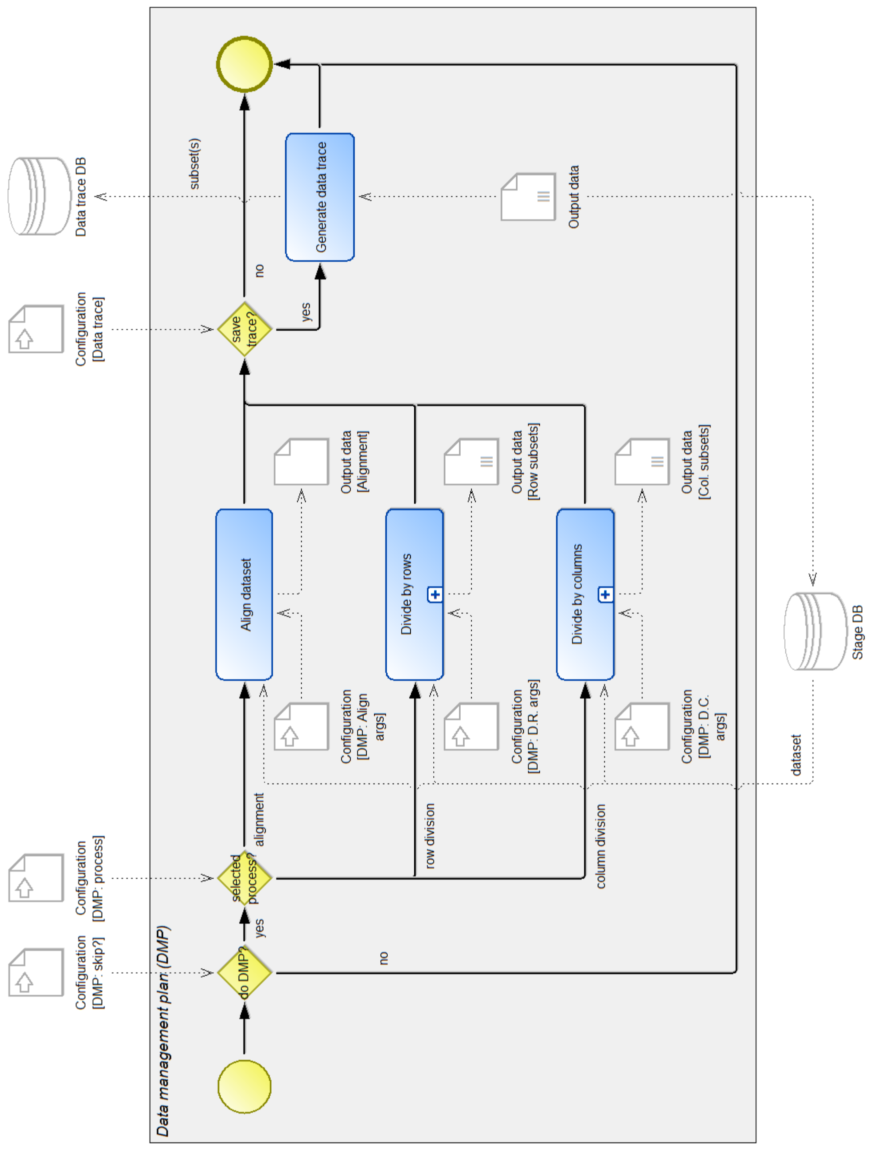


Figure 5.4: Design of PhyloFlow's fetching stage in BPMN. The dataset or datasets can be fetched from both local files and public databases, like GenBank.

module of Biopython. We have also included a few predefined queries of specific biological types of data for inexperienced users. For example, the user can introduce *human mtDNA* as keyword in the configuration process and the system will access to GenBank and fetch all the resultant sequences of the following query: *"homo sapiens"[Organism] AND mitochondrion[All Fields] AND "complete genome"[Title] NOT chromosome[All Fields] NOT mRNA[Filter].*

Generating a good data management plan has a double purpose: to establish the best feasible foundation to infer an accurate phylogeny for the given input dataset, and to improve the performance of the system. Hence, the second stage has a high impact on the system's capability to provide a completely adaptable configuration. Its design, presented in Figure 5.5, reflects the flexibility achieved both in the aforementioned performance aspects and in the application of biological knowledge. There are three main procedures included in this stage that can be selected iteratively multiple times. The first one is the MSA, which will be applied individually to each input dataset. In the current version of PhyloFlow, we have included MAFFT [29], Clustal Omega [45] and MUSCLE [46]. The second procedure deals with the row division, where each dataset is divided into subsets of full sequences, following different criteria: a naïve division, a biological division, that is, a division guided by biological knowledge, e.g. haplogroup clustering for hmtDNA sequences, and an overlapping division. The latter is achieved through PRD, as we did with MEvoLib. The last procedure is concerned with the column division, where each dataset is divided into subsets of fragments from all the sequences. As in the previous case, diverse criteria can be applied to perform this process: a naïve division and a biological division. For instance, the latter could be a gene division for DNA sequences. The second and third procedures represent a divide-and-conquer strategy, enabling the application of current tools used in the next phases, or improving their time cost. Moreover, if the system is going to be executed in a multicore/multithreaded platform, this



Data management plan (DMP)
do DMP?
yes
no
selected process?
alignment
row division
column division
save trace?
yes
no
Align dataset
Divide by rows
Divide by columns
Generate data trace
Configuration [DMP: skip?]
Configuration [DMP: process]
Configuration [DMP: Align args]
Configuration [DMP: D.R. args]
Configuration [DMP: D.C. args]
Configuration [Data trace]
Output data [Alignment]
Output data [Row subsets]
Output data [Col. subsets]
Output data
subset(s)
dataset
Data trace DB
Stage DB

Figure 5.5: Design of PhyloFlow's data management plan stage in BPMN. The datasets fetched can be aligned or splitted in subsets of sequences (e.g. haplogroups) or fragments (e.g. genes).

data slicing process will improve the total time cost trough parallelization. Additionally, this stage can be completely skipped for specific scenarios, e.g. when the phylogenetic estimation process does not require an MSA [60].

The third stage focuses on the phylogeny estimation. In the current version of PhyloFlow, we have focused on ML methods due to its extensive use nowadays, the impractical time cost of the Bayesian inference methods for large datasets [98] and the accuracy and statistical downsides of parsimony and some distance-based methods. Besides, a special attention to include *model selection* in ML has been considered, based on the hypothesis that it can achieve a deep impact on the accuracy improvement for the phylogenetic estimation process. It is important to highlight that using the same model for every input dataset may not produce an accurate phylogeny. On the other hand, testing every existing evolution model for each input dataset can be highly intensive in time cost. In order to guarantee statistical robustness, a bootstrapping procedure can be included within the inference process. The complete design of this stage is displayed in Figure 5.6. This stage provides three different tools for ML: PhyML [77], RAxML [79] and FastTree [81]. The first and second tools can be used to explore a broad selection of the best evolution model, whilst the third one has been included for its widespread use and low computation costs. Apart from using the bootstrapping option included in some of those software tools, PhyloFlow incorporates Seqboot from PHYLIP [100]. We made a minor source code modification to control the path creation of the output file, making it fully compatible with our framework.

As shown in Figure 5.7, the fourth stage will assemble all the previously obtained phylogenies. This procedure can be completely skipped or repeated several times iteratively. The former is intended for workflows where the previous stage has generated one single phylogenetic tree as output, whilst the latter covers the scenarios where more than one division has been performed at the data management plan stage. The configuration process has

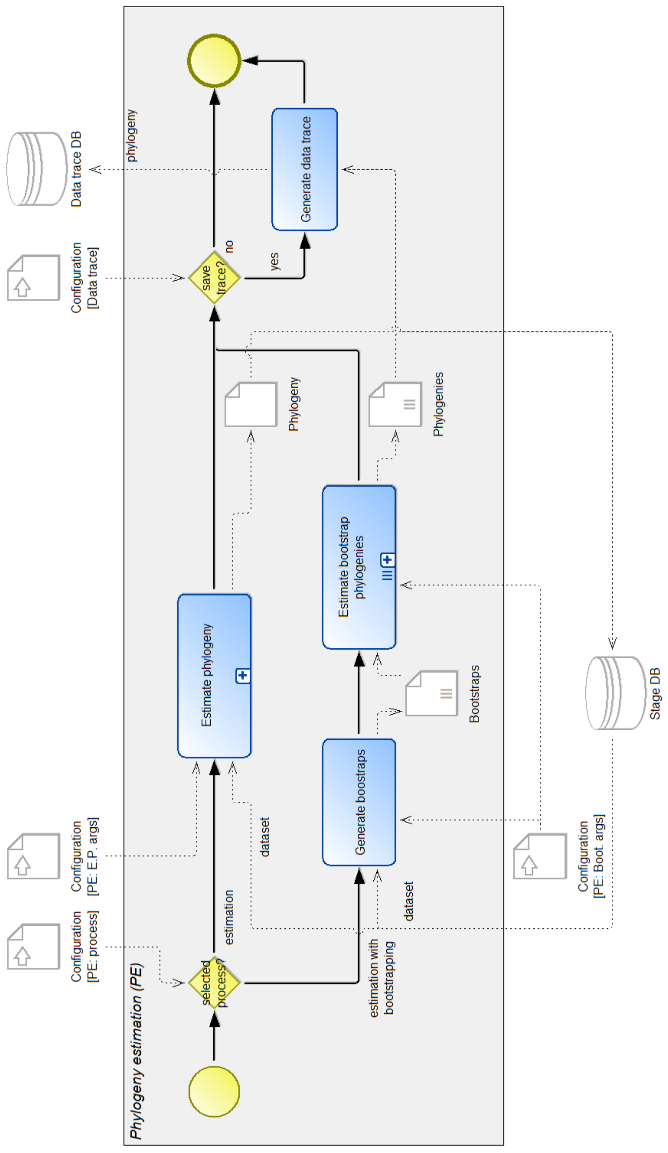

Phylogeny estimation (PE)
selected process?
estimation
estimation with bootstrapping
Configuration [PE: process]
Configuration [PE: E.P. args]
dataset
Estimate phylogeny
Generate boostraps
dataset
Configuration [PE: Boot. args]
Bootstraps
Estimate bootstrap phylogenies
Stage DB
Phylogeny
Phylogenies
save trace?
yes
no
Configuration [Data trace]
Generate data trace
phylogeny
Data trace DB

Figure 5.6: Design of PhyloFlow's phylogenetic estimation stage in BPMN. The phylogenetic inference procedure can be applied with or without bootstrapping.

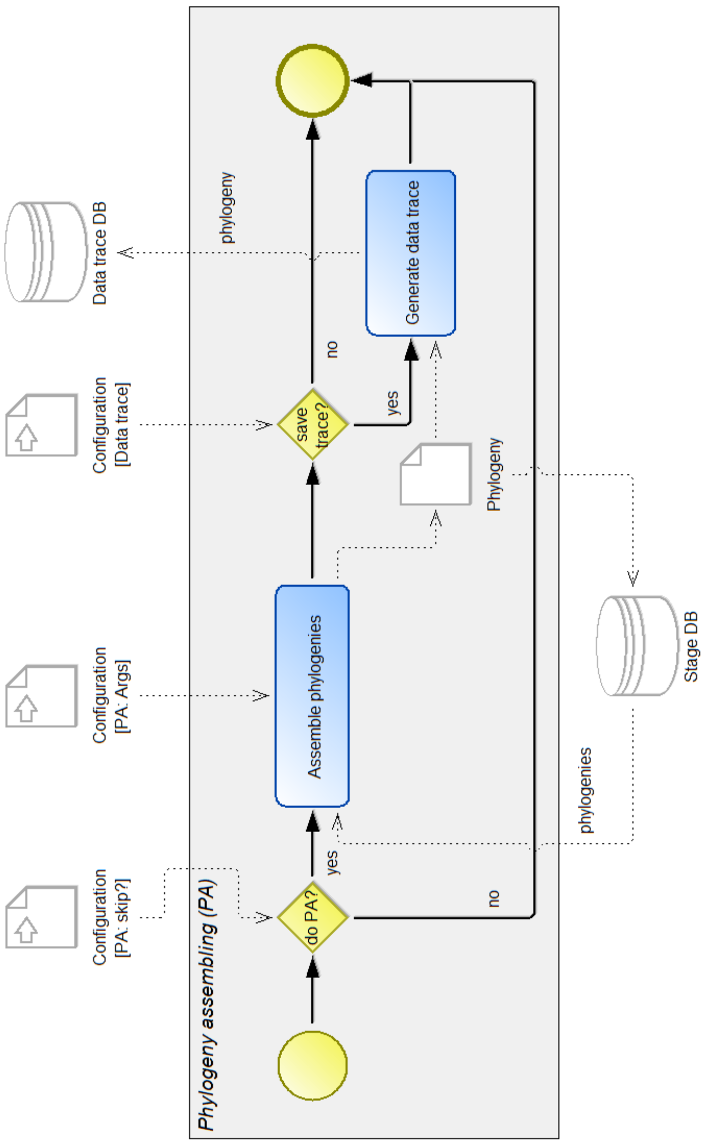


Figure 5.7: Design of PhyloFlow's phylogenetic assembling stage in BPMN. The procedures available in this module are suitable only for those workflows where a data division or a bootstrapping have been performed in previous stages.

been programmed to offer only viable options within the processes selected so far. For instance, the user will not be able to build a workflow with a phylogenetic assembling process if the previous stage is going to output a single phylogenetic tree. The implementation of the fourth stage of PhyloFlow bears on diverse popular methods to assemble phylogenies. The first one is the consensus method. This procedure is achieved through Consense from PHYLIP. As it happened with Seqboot in the previous stage, this tool has also been modified to have more control on the creation of the output file. The second method is SuperFine, for supertree constructions. The third alternative is the assembling method we published in 2011 [161], which aims to build a phylogeny from non-overlapping phylogenetic trees based on a topology profile. The key idea of this method is to take advantage of the biological knowledge of well-known relationships of clusters of certain types of data. Usually, their affinity is drawn in a phylogenetic tree (profile tree). Hence, the leaf of each cluster in the profile tree can be replaced by its matching phylogeny to obtain the final supertree. This process is similar to the final stage of ZARAMIT.

The last stage of PhyloFlow has been designed to group the subsequent analysis and mining of the resultant phylogeny, such as the incorporation of the analysis software tools we will present in Chapters 6, 7 and 8.

5.3.2 Workload characterization

We developed a preliminary evaluation of the methods included in the current version of PhyloFlow. For such purpose, we generated several datasets with random complete hmtDNA sequences from GenBank. The initial estimations obtained can be used for a latter evaluation of the scalability of any customized system. The computer used was an Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz with 16G DIMM DDR3 1600 MHz.

The characterization of the workload shown in Tables 5.1, 5.2, 5.3 and 5.4 is based in terms of procedures, execution time and memory usage. It outlines the complexity stemming from the inherent features of the input data (number of sequences, length variability and heterogeneity) together with the wide pool of tools applicable along the inference workflow, especially if we consider the wide range of parameters available for each tool.

Table 5.1: Workload characterization of the fetching stage.

Procedure	Fetch hmtDNA seqs. from GenBank			
Num. Seqs.	10	100	1000	10000
Time(s)	2.44	4.27	39.66	334.18
Mem.(MB)	28.06	40.14	160.68	3114.63

5.3.3 Practical applications

We designed PhyloFlow to achieve two main objectives: **i)** to grant the recreation of other existing phylogenetic inference systems; and, **ii)** to prove its capability to work even with very large input datasets. As we will show below, PhyloFlow generates an enhanced version of the original system, as it will include the customization and operability under different computer architectures offered by our framework. The latter objective will be proved generating a workflow based on the procedure followed by hmtDNA experts in ZARAMIT. As we have already stated, the hmtDNA is a real and challenging large-sized problem.

Recreation of an existing system: *DACTAL*

We have selected DACTAL as template to prove the recreation capabilities of PhyloFlow. After fetching the desired dataset, DACTAL uses MAFFT `--parttree`, a performance-oriented MSA method, and FastTree to quickly generate the initial phylogenetic tree required to start its iterative procedure. Next, the PRD process is applied to this initial phylogenetic tree and the

Table 5.2: Workload characterization of the data management plan stage. *Hgs.* stands for *haplogroups*.

Procedure	MSA			Col. division		Row division		
Configuration	*MAFFT* `--auto`	*Clustal Omega*	*MUSCLE*	*Naïve* 100 sets	*Genes*	*Naïve* 10 sets	*Hgs.*	*PRD* 25 & 4
Num. Seqs.	400			1000	100	1000	10	100
Time(s)	27.38	26133.06	2480.37	8.08	10.58	2.12	228.85	294.12
Mem.(MB)	428.49	1710.04	4021.31	432.46	19.08	150.00	14.45	55.10

Table 5.3: Workload characterization of the phylogenetic estimation stage.

Procedure	Bootstrapping	ML		
Configuration	*Seqboot* 10 bootstraps	*PhyML* GTR+CAT	*RAxML* GTR+CAT	*FastTree* GTR+CAT
Num. Seqs.	100	10		
Time(s)	2.88	75.38	1.47	4.09
Mem.(MB)	16.03	10.58	4.30	9.76

Table 5.4: Workload characterization of the phylogenetic assembling stage.

Procedure	Consensus	Profile tree	Supertree
Configuration	*Consense*	*Haplogroups*	*SuperFine*
Num. Seqs.	100		
Phylogenies	10		
Time(s)	0.12	1.41	4.07
Mem.(MB)	10.78	11.12	17.59

subsets of sequences are aligned again, this time with MAFFT `--auto`. Later on, each MSA is used to infer its phylogeny. Finally, the final phylogenetic tree is estimated by SuperFine. As we have aforementioned, PhyloFlow includes all the software tools required to make an exact replica of this workflow. The meta-script would be in charge of the iterative refinement process. Additionally, this reproduction can be customized, changing either the method or the parameters used at any step. Advanced and expert users could improve significantly the accuracy of the resultant phylogeny taking advantage of this added feature, e.g. picking a more realistic evolution model in the inference stage for the biological data fetched.

Phylogenetic inference on very large scenarios: *hmtDNA*

Concerning the hmtDNA phylogenetic inference study, the first step is to fetch all the hmtDNA biological sequences to get the most updated result. Thus, the workflow would start with a fetching process from GenBank using the predefined query for complete hmtDNA sequences. Based on the workload presented on Table 5.1 and the number of sequences at GenBank shown in Figure 1.3, this process would take roughly 20 minutes.

Applying the biological knowledge available for hmtDNA, the input dataset can be divided into 35 subsets corresponding to the main haplogroups. Afterwards, each subset can be separated into 38 subsets with a gene splicing based on its genetic code. Hence, this multiple division procedure would end up with 1330 (35×38) datasets which must be aligned before moving to the next stage. The estimated sequential cost of the aforementioned process based on Table 5.2 is 8 days. The execution time related to this stage can be significantly reduced by taking advantage of the inner parallelism of the disposed architecture, making a first division of the input dataset into one subset per available core. Both distributions can be performed separately. The 1330 MSAs must be computed after the fragments are rearranged in their corresponding files. The results of this stage will probably have a negative impact in the performance of the whole system due to their unbalanced nature. For instance, the smallest haplogroup has less than 10 sequences, whilst the largest possesses more than 4000. Something similar happens with the gene lengths (70 vs 1800 nucleotides).

For the phylogenetic estimation stage, we can use the same 88 evolution models of the first version of jModelTest [92] to include a model evaluation. The number of bootstraps can be set up to 200, a good value in the recommended range of 100 to 1000. New problems arise at this point. The first one is concerned with the amount of processes to be created: for the model evaluation with PhyML, we would have 117040 tasks, and for the bootstrapping step, the system would generate 266000 processes. The second problem is derived from the first one: the total time required to execute the whole stage. Using the workload characterization of this stage at Table 5.3, the sequential execution time of the whole stage would be about 2078 years. In addition, PhyML does not work with datasets constituted by more than 4000 sequences; hence, some haplogroups would require an additional data management plan processing beforehand. Finally, the phylogeny of each haplogroup is merged applying our *profile tree* process at the phylogeny assembling stage, which would take about 7 minutes (Table 5.4). The phylogenetic profile incorporated to PhyloFlow has been extracted from PhyloTree, keeping only the tree composed by the 35 main haplogroups.

5.4 PHYSER

PHYSER [126] is a phylogenetic inspired algorithm to assess the quality of new sequences by their location in the reference tree. As a byproduct, we can use PHYSER to update the phylogenetic tree with new published sequences, thus offering a good compromise between accuracy and an up-to-date state of the phylogeny between its global updates. The algorithm provides the classification or qualitative value of the input sequence, the total number of differences that are found between the closest node of the phylogeny and the input sequence, i.e. *distance*, and the list of possibly erroneous sites.

5.4.1 Design and implementation

PHYSER draws inspiration from the expert procedures applied to the incremental construction of MITOMAP's phylogeny [34]. The algorithm is based on the fact that we are not able to determine whether a mutation is real or not just by looking at the sequence to which it belongs. The evolution knowledge can be very helpful to assess the quality of a sequence. A phylogenetic tree encloses a lot of information to deduce how a specific sequence might have evolved, being straightforward to verify the new sequence regarding its whereabouts in the phylogeny. Undoubtedly, an accurate and truthful phylogenetic tree is vital to obtain reliable results. We assume that every given phylogeny meets this condition. Thus, PHYSER locates the place in the phylogeny where the input sequence fits better and then, it evaluates the quality of the sequence with the information of its neighborhood sequences. Algorithm 5 shows the details of its implementation.

Algorithm 5: PHYSER algorithm

```
Data: $s$ (new sequence), $r$ (reference sequence), $T$ (phylogeny)
Result: $v$ (evaluation of $s$), $\hat{T}$ (updated phylogeny)
1  root$(T) \longleftarrow N_i$;
2  while $\neg leaf(N_i) \wedge (N_{i-1} \neq N_i)$ do
3      $N_i \longleftarrow N_{i-1}$;
4      reference_filter$(s, r, T, N_i) \longleftarrow N_i$;
5      if $N_{i-1} == N_i$ then
6          hamming_filter$(s, T, N_i) \longleftarrow N_i$;
7      end
8  end
9  reference_distance$(s, N_i) \longleftarrow d$;
10 if $d \leq threshold_1$ then
11     "RIGHT" $\longleftarrow v$;
12 else if $d \leq threshold_2$ then
13     "CAUTION" $\longleftarrow v$;
14 else
15     "WRONG" $\longleftarrow v$;
16 end
17 Set $s$ as sibling of $N_i$ in $T \longleftarrow \hat{T}$;
```

The *Reference filter* computes the *reference distance* (Algorithm 6) for the sequence at the given node N_i and all its children nodes. The method returns the node or nodes with the lowest distance.

The *Hamming filter* calculates the *hamming distance* (Algorithm 7) for the sequence at the given node N_i and all its children. This method returns

Algorithm 6: Reference distance

```
Data: $s$ (new sequence), $r$ (reference sequence), $s_T$ (sequence from $T$)
Result: $d$ (reference distance between $s$ and $s_T$)
1  align($s_T, r, s$);
2  len($s_1$) $\longleftarrow m$;
3  empty_list $\longleftarrow l$;
4  for $i \in \{1, .., m\}$ do
5      if $r[i] \neq$ gap $\wedge$ $r[i] \neq s_T[i]$ then
6          $l$.append($i$);
7      end
8  end
9  0 $\longleftarrow d$;
10 for $j$ in $l$ do
11     if $s[i] \neq s_T[i]$ then
12         $d{+}{+}$;
13     end
14 end
```

the node or nodes with the lowest distance. It is applied when only the *parent* node has been returned by the reference distance. We refer to this situation as *local minima*, i.e. the parent results as the closest node, but it is just a local situation: there are other nodes, nearer to the leaves of the phylogeny, which are more related to the new sequence than the parent is. The filter can return the parent again as the closest node. In this case, we have reached a *global minimum*, so it is the closest node to the input sequence of the entire phylogenetic tree.

In the majority of the comparisons, one of the *children* nodes will be the closest to the new sequence. However, there are other situations that we have taken into account but they are not shown in Algorithm 5. The first two cases refer to the situation when more than one node have the lowest distance. In this case, we first remove the *parent* node from the list inasmuch we prefer to get closer to the leaves. The statistical significance of the results

Algorithm 7: Hamming distance

```
**Data:** $s_1, s_2$ (sequences)
**Result:** $d$ (hamming distance between $s_1$ and $s_2$)
align($s_1, s_2$);
len($s_1$) $\longleftarrow m$;
$0 \longleftarrow d$;
for $i \in \{1, .., m\}$ do
    if $s_1[i] \neq$ gap $\wedge\ s_2[i] \neq$ gap $\wedge\ s_1[i] \neq s_2[i]$ then
        $d{+}{+}$;
    end
end
```

of our algorithm increases as more comparisons we perform, that is, as we explore more nodes of the phylogeny. Once the *parent* has been removed, if we still have more than one solution, the algorithm will explore each new path independently, applying the main process individually. The tests we will show later on demonstrate that this "multipath" situation will not last longer than two or three iterations in most cases. Intuitively, the algorithm may display more than one node as solution. When this happens, all the proposed solutions are usually close relatives, that is, nodes with the same parent node or nephew nodes.

The algorithms uses two thresholds to determine if the sequence is *Right*, if the user should proceed with *Caution* due to the detection of plausible errors, or if the sequence is most probably *Wrong*. It is important to remark that these thresholds will not work properly if the new sequence corresponds to any unexplored species within the phylogeny, or other similar cases, which are depicted as sparsely populated branches.

5.4.2 Evaluation

Although PHYSER can work with any kind of data, we chose hmtDNA data for the algorithm testing. For our experiments we used the last phylogenetic tree created by ZARAMIT project [26], which is composed by 7390 complete hmtDNA sequences obtained from GenBank. As the reference sequence, we selected the rCRS. Dr. Ruiz-Pesini, a biologist expert in mtDNA, suggested to set the thresholds to 0 for the *Right* - *Caution* discrimination, and 3 for the *Caution* - *Wrong* distinction. All the experiments were executed in a computer with a Intel(R) Core(TM) 2 Duo Processor E6750 @ 2.66GHz and 8GB RAM. The phylogenetic tree and the information needed by the application required less than 30 seconds to load. They just have to be loaded the first time, when the application initiates. The program took 12 seconds on average (with a standard deviation of 1.4) to locate the input sequence. Hence, the program has an excellent performance and the user can obtain the results in “real time”, providing an accurate feedback showing the sites that have been checked as bad with the closest nodes.

We arranged the first experiment to test the accuracy of the algorithm. We selected the 62 sequences published in [167] (AY738940 to AY739001) and the 23 sequences reported in [168] (DQ246811 to DQ246833). All these identifiers correspond to leaves of the chosen phylogenetic tree. As result, the algorithm classified all of them as *Right*, which implies a success rate of 100%. However, only 53 were located correctly, i.e. a 60.95% of accuracy. In all but 2 of the unspotted sequences, the returned nodes were close relatives. Sequences DQ246830 and DQ246833 returned an unexpected *distance* value of 273, where the average for the rest of the sequences was 9 with a standard deviation of 5. It turned out that these two sequences are 249 nucleotides shorter than the rCRS. Therefore, the inability of the algorithm to locate those sequences was disregarded.

The second test was designed to study the behavior of PHYSER when the biological sequences do not belong to the given phylogenetic tree. In this case, we chose several animal mtDNA sequences that are displayed together with the results of this experiment in Table 5.5. Before the classification, each sequence was aligned with the rCRS using MUSCLE. Afterwards, we deleted in both sequences all those sites that correspond to gaps introduced by the MSA tool in the rCRS. This process is considered to prevent any degradation to the relevant phylogenetic information of the sequence. Notice that the main purpose of PHYSER is to detect sequencing errors. Thus, the fact that all sequences were classified as *Right* is not a drawback of our algorithm but more a consequence of the closeness of those species to the *Homo sapiens sapiens* (regarding the mtDNA). The most relevant result drawn from this experiments is the *distance* field. This value is provided altogether with the classification due to its relevance in a good interpretation of the output. Of all the animals, the closeness of chimpanzee does not seem surprising. Besides, it is possible to see the relationship with the different classes and clusters as far as we go deeper in the evolution process. In most cases, the distance implies more than a 30% of wrong nucleotides, a good indicative of a different problem from the one we are trying to solve here.

Table 5.5: PHYSER classification of diverse animal mtDNA sequences.

Accession	Animal	Classification	Distance
NC_001643	Chimpanzee	RIGHT	1966
NC_001941	Sheep	RIGHT	4719
NC_005313	Bullet tuna	RIGHT	5775
NC_009684	Mallard duck	RIGHT	5818
NC_007402	Sunbeam snake	RIGHT	6322
NC_002805	Dark-spotted frog	RIGHT	6191
NC_008159	Mushroom coral	RIGHT	8242
NC_009885	Nematode	RIGHT	9074
NC_006281	Blue crab	RIGHT	9251
NC_006160	Whitefly	RIGHT	9302

The third experiment aimed to analyze the capacity of the algorithm to detect single relevant mutations. We created five synthetic sequences from the sequence AY738958 introducing diverse mutations manually. The mutations and classification of each sequence are shown in Table 5.6, together with the classification of the sequence AY738958 as reference. The first three mutated sequences demonstrate how a relevant mutation can change the classification from *Right* to *Caution*. Furthermore, if we disrupt the sequence to certain level, the closest node can also change, as it happens with the two last mutated sequences. In most cases, the result will change from one node to one of its close relatives, as it happened in our experiment. Nevertheless, the mutations or sequencing error can affect key sites of the sequence, changing the closest node significantly.

5.5 Conclusions

In this chapter, we have presented PhyloFlow, the first fully customizable and automatic framework to design and deploy phylogenetic inference and analysis workflows. As its main characteristics, PhyloFlow provides large dataset tractability, adaptability to user's requirements and accuracy's enhancement by biological-driven methodologies. Furthermore, the resultant systems will run automatically, i.e. unattended execution. Our framework incorporates a novel interactive configuration process to compose the workflow, comprising its software tools and parameterizations. It integrates well-known programs and methodologies for each usual phylogenetic analysis process. Additionally, PhyloFlow includes a guideline to help non-expert users to customize their own systems. Its design follows a divide-and-conquer strategy along with a black-box principle of transparency for a full compatibility within software tools. As result, PhyloFlow can assemble systems that perform efficiently even with large datasets. We have studied the workload characterization of PhyloFlow using a representative set of hmtDNA sequences and a reduced set of parameters for the tools. This is a complex task due to the variety of workloads induced by the number of sequences, their sizes and their het-

Table 5.6: PHYSER classification of diverse synthetic sequences created from AY738958. Each mutation is depicted by the scheme: *previous value* + *site* + *new value*. The number in parenthesis after the classification shows the distance to the surpassed threshold.

Accession	Mutation	Classification	Closest nodes	Distance
AY738958	*base sequence*	RIGHT	Anc3564, Anc4104	7
SEQ00001	-3106A	RIGHT	Anc4104	6
SEQ00002	-3106A, G8859A	CAUTION (1)	Anc4104	7
SEQ00003	-3106A, G8859A, G15325A	CAUTION (2)	Anc4104	8
SEQ00004	T6775C	RIGHT	Anc4076, Anc4104	7
SEQ00005	T6775C, G1437A	RIGHT	EU130575	13

erogeneity. These results help to estimate the balance and time cost of the configured workflow. Besides, we have demonstrated the capability of our system to reproduce other popular phylogenetic inference systems with the extended capability of tool selection and full parameterization. Additionally, the strengths or these systems are enhanced with the incorporation of tasks normally applied before and after them, e.g. dataset fetching or further analyses of the results. The hmtDNA workflow has shown an application of more specific methods and procedures, like model selection, that incorporates biological knowledge to phylogenetic analysis systems.

The huge time and economical expenses of inferring large phylogenies force an update every long periods. For instance, the hmtDNA phylogeny we have studied with PhyloFlow would be usually inferred at most once a year, as it happens with PhyloTree. On the other hand, new biological sequences are uploaded to public databases at a rate of tens or hundreds a month. This contrast pose a delicate situation on phylogenetic studies, were new information is discarded until a new phylogeny is inferred. Moreover, the validation of those new sequences is still made by hand, implying a large investment of time, disregarding possible human mistakes. PHYSER is a new algorithm we have developed to assess sequencing errors relying on phylogenetic information. Additionally, our method updates the phylogeny used as reference with new evaluated sequences, providing a convenient solution in between phylogenetic updates. Our tests show that PHYSER comprises an accurate detection of sequencing errors together with an excellent time cost: only a few seconds to evaluate and incorporate a complete mtDNA sequence in the phylogenetic tree.

As future work, we plan to renovate PhyloFlow's multiplatform support of different hardware architectures. We also aim to replace the question about the hardware platform in the configuration process for an internal task that will adjust the workflow produced to the available hardware platform before launching it. Furthermore, we want to redesign PhyloFlow's source code to take advantage of MEvoLib, so new software incorporations in the latter can be applied directly in the former. We target to improve PHYSER with a new classification of the possible errors detected including a new criterion of biological viability. It will consist of taking into account the reversions, i.e. mutations that have appeared before in the path we have explored in the phylogeny. They usually imply a more probable and harmless change in the biological sequence. Moreover, the conservation analysis of different species will provide an extra criterion to the biological viability of the mutation. We will introduce this analysis in the next chapter.

Everybody knows what a caterpillar is,
and it doesn't look anything like a butterfly.

Lynn Margulis

6 Phylogenetic analysis: *Display of large phylogenies*

The first phylogenetic analysis we are going to introduce in this dissertation is the study of those aspects related to the molecular evolution of the selected sequences available in the phylogeny inferred. Biologists usually achieve this objective through visualization software tools that allow them to interact with the phylogenetic tree. The results presented in this chapter are the outcome of the undergraduate project of Samuel Guajardo [169].

In the first section of this chapter, we cover the state-of-the-art of phylogeny visualization tools. Afterwards, we present our contribution to the field with a new software tool that includes novel features. Finally, the last section assembles the conclusions for this chapter. The results in this chapter have been presented in an international conference [170].

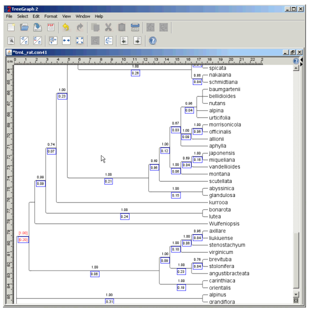


Figure 6.1: Screenshot of TreeGraph 2 provided in its tutorial (*http://treegraph.bioinfweb.info/Help/wiki/Tutorial:Main_page*).

6.1 Related work

The visualization of phylogenetic trees is a fundamental part on the analysis performed by biologists. Providing an interface capable of meeting the requirements of navigability and information display are essential to the development of a successful visualization tool. Thus, many software tools have been developed to fulfill different aspects of the phylogenetic visualization: TreeGraph [171], FigTree [172] or Dendroscope [2], among others. Furthermore, many web-oriented platforms have been created to provide a similar approach but with the addition of *visibility* and public usage of the uploaded phylogenies. Some examples are iToL (*interactive* ***T****ree* ***o****f* ***L****ife*) [173], EvolView [174], PhyloWidget [3] or PhyloTree [41].

Most of the aforementioned tools provide the standard navigation aspects: a general display of the whole phylogeny with zoom-in and zoom-out options, branch lengths can be show or hidden and further information is displayed when clicking in the corresponding identifier of each biological sequence. If the phylogeny is larger than the screen resolution, a scroll bar will be enabled at the side of the window to visualize the entire tree. Besides, the majority of the tools accept as input the NEWICK tree format, among others. Figures 6.1, 6.2, 6.3, 6.4, 6.5, 6.7 and 6.6 show interface examples of each tool listed earlier.

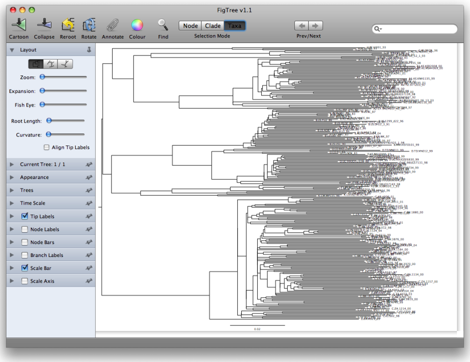


Figure 6.2: Screenshot of FigTree provided at its website (*http://tree.bio.ed.ac.uk/software/figtree/*).

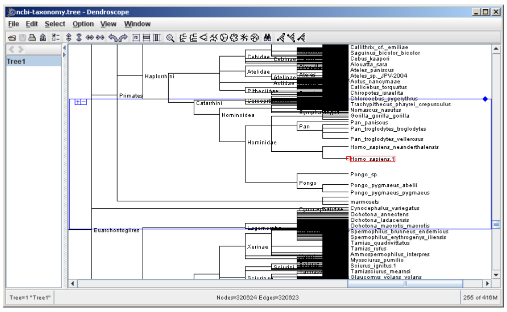
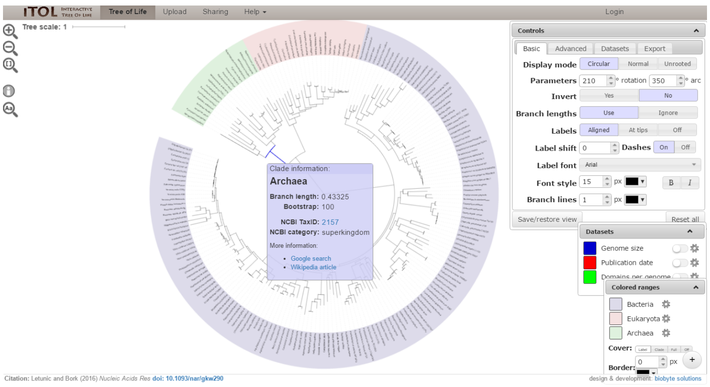


Figure 6.3: Screenshot of Dendroscope provided at its website (*http://dendroscope.org/*).


iTOL Interactive Tree Of Life
Tree of Life
Upload
Sharing
Help
Login
Tree scale: 1
Controls
Basic
Advanced
Datasets
Export
Save/restore view
Reset all
Datasets
Colored ranges
Citation: Letunic and Bork (2016) Nucleic Acids Res doi: 10.1093/nar/gkw290
design & development: biobyte solutions

Clade information:
Archaea
Branch length: 0.43325
Bootstrap: 100
NCBI TaxID: 2157
NCBI category: superkingdom
More information:
- Google search
- Wikipedia article

Display mode: [✓ Circular] Normal Unrooted
Parameters: 210° rotation 350° arc
Invert: Yes [✓ No]
Branch lengths: [✓ Use] Ignore
Labels: [✓ Aligned] At tips Off
Label shift: 0 Dashes: [✓ On] Off
Label font: Arial
Font style: 15 px
Branch lines: 1 px

Datasets:
🟦 Genome size
🟥 Publication date
🟩 Domains per genome

Colored ranges:
▨ Bacteria
▨ Eukaryota
▨ Archaea
Cover: Label Clade Full Off
Border: 0 px

Figure 6.4: Screenshot of iToL.

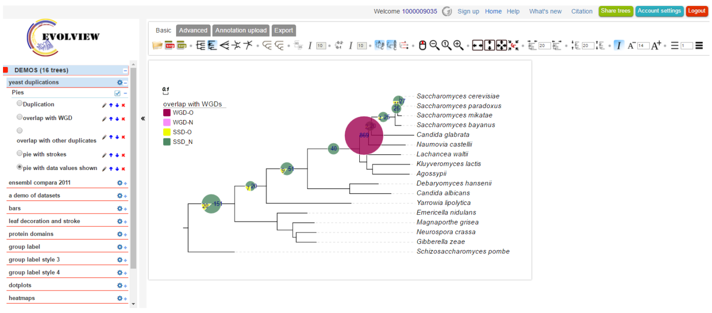
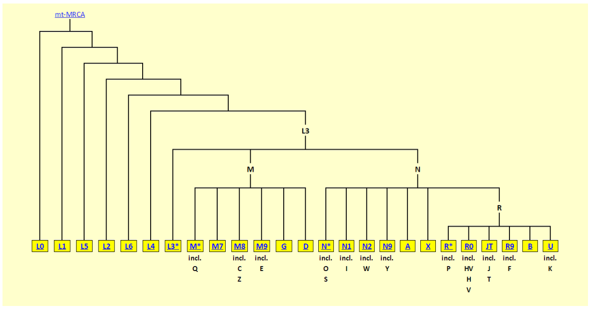


Figure 6.5: Screenshot of EvolView.

(a)



(b)

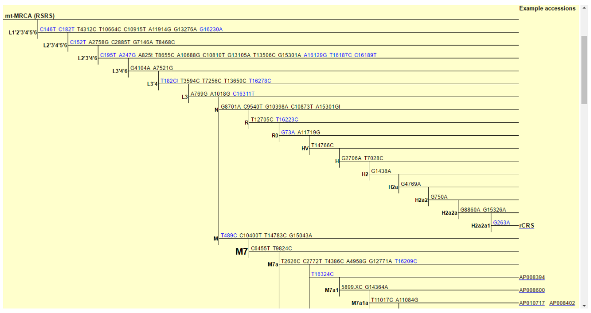
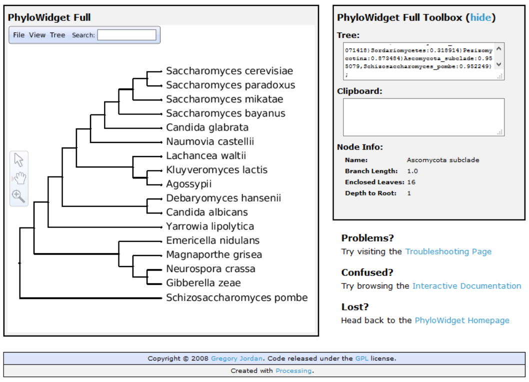

mt-MRCA (RSRS)
Example accessions
L1'2'3'4'5'6 C146T C182T T4312C T10664C C10915T A11914G G13276A G16230A
L2'3'4'5'6 C152T A2758G C2885T G7146A T8468C
L2'3'4'6 C195T A247G A825t T8655C A10688G C10810T G13105A T13506C G15301A A16129G T16187C C16189T
L3'4'6 G4104A A7521G
L3'4 T182C! T3594C T7256C T13650C T16278C
L3 A769G A1018G C16311T
N G8701A C9540T G10398A C10873T A15301G!
R T12705C T16223C
R0 G73A A11719G
HV T14766C
H G2706A T7028C
H2 G1438A
H2a G4769A
H2a2 G750A
H2a2a G8860A G15326A
H2a2a1 G263A rCRS
M T489C C10400T T14783C G15043A
M7 C6455T T9824C
M7a T2626C C2772T T4386C A4958G G12771A T16209C
T16324C AP008394
M7a1 5899.XC G14364A AP008600
M7a1a T11017C A11084G AP010717 AP008402

Figure 6.6: Screenshots of PhyloTree: **a)** haplogroup phylogeny; **b)** detailed phylogeny of the haplogroup M7, including paths from the root to the rCRS and to the selected haplogroup.



```
071418)Sordariomycetes:0.318914)Pezizomy
cotina:0.873484)Ascomycota_subclade:0.95
5079,Schizosaccharomyces_pombe:0.952249)
;
```

Name: Ascomycota subclade
Branch Length: 1.0
Enclosed Leaves: 16
Depth to Root: 1

Problems?
Try visiting the Troubleshooting Page

Confused?
Try browsing the Interactive Documentation

Lost?
Head back to the PhyloWidget Homepage

Copyright © 2008 Gregory Jordan. Code released under the GPL license.
Created with Processing.

Figure 6.7: Screenshot of PhyloWidget.

TreeGraph 2 [175], displayed in Figure 6.1, includes a graphical editor that allows the creation and modification of phylogenetic trees. Another example of a visualization software tool with additional features is PhyloTree. As we have mentioned in the previous chapter, it is a web-based hmtDNA's haplogroup phylogenetic tree. In Figure 6.6 we can see that the whole phylogeny is represented by the hmtDNA haplogroup tree (image *(a)*). Once a specific haplogroup is chosen, the view is changed to the structure shown in image *(b)*, where the scroll bar and search functionality of the internet explorer are the unique navigation features provided. In this tool, the text of each branch has the list of mutations that defines that subtree instead of the typical branch length or the consensus value. The phylogeny does not contain all the available complete hmtDNA sequences at the date it was inferred; only one sample is given from those that have all the corresponding subtree mutations.

However, real life exploitation of the results obtained with inferred large phylogenies is limited by the lack of specific tools able to provide enhanced navigation and search features. To our knowledge, there are no specific tools prepared to cope with phylogenetic trees of more than a few thousand nodes. Moreover, no one can link additional biological data in terms of structural features, metadata or fine grain details about the biological sequences in the

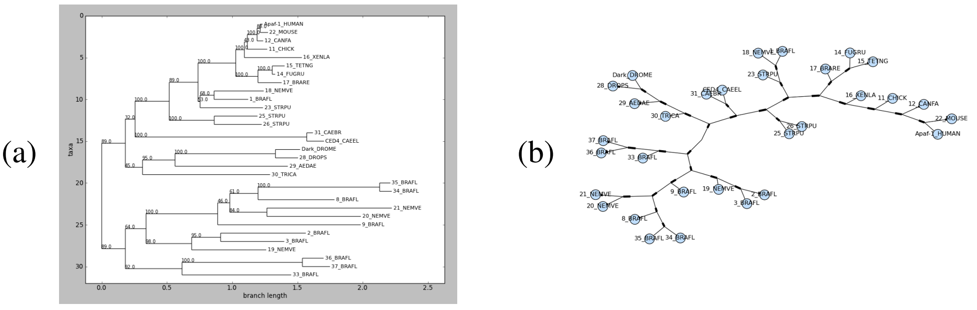


Figure 6.8: Screenshot of two phylogenies displayed with two different methods of the *Phylo* module of Biopython: **a)** `draw()` method (using Matplotlib [176] library); **b)** `draw_graphviz()` method (using Matplotlib [176], NetworkX [177] and PyGraphviz [178] libraries). Both screenshots are provided at *Phylo* module's website (*http://biopython.org/wiki/Phylo*).

nodes. Only PhyloTree is able to handle a two-level granularity of the phylogenetic information, providing a haplogroup representation and a sequence phylogeny of each haplogroup.

Additionally, standalone and web-based software programs are usually avoided in the design of workflows and scripts due to their user-oriented interface. Thus, other alternatives with less potential are selected in these scenarios. For instance, Biopython [13] offers three phylogenetic display functions within its *Phylo* class. The two graphical alternatives are shown in Figure 6.8. The third method is `draw_ascii()`, which "draws" the phylogenetic tree in the command-line terminal using only ASCII characters. The style of the text-based phylogeny would be similar to the image *(a)* displayed in Figure 6.8.

Under these premises, we have developed a new phylogenetic visualization software tool named PhyloViewer.

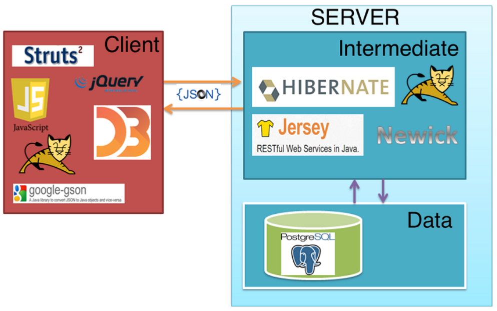


Figure 6.9: Client/Server architecture of PhyloViewer, including all the technologies used in each part.

6.2 PhyloViewer

PhyloViewer [170] is a visualization tool specifically developed to study large phylogenetic trees. It has been designed and implemented based on the concept of usability. PhyloViewer consists of a client/server architecture based on a web interface and a database server. Figure 6.9 shows the details of the architecture and the technologies used at each part. Notice that all the components of PhyloViewer are *open source*.

The design of the interface was designed following the requirements provided by a group of biologists specialized in the field of mtDNA, where most of the phylogenies usually exceed the “thousand nodes” threshold. Furthermore, we were interested in the display of a differentiated, more functional structure of the phylogenetic tree. We set the groups, or haplogroups in the case of hmtDNA, and the data associated to each node with a more relevant representation than the whole phylogeny. This was also motivated by the impossibility of illustrating a large phylogeny clearly. Thus, PhyloViewer has an interactive navigation over a main panel which provides a father-siblings view, as it is shown in Figure 6.10. Besides, each biological sequence includes a complementary visualization with user annotations and additional data (e.g. GenBank’s metadata).

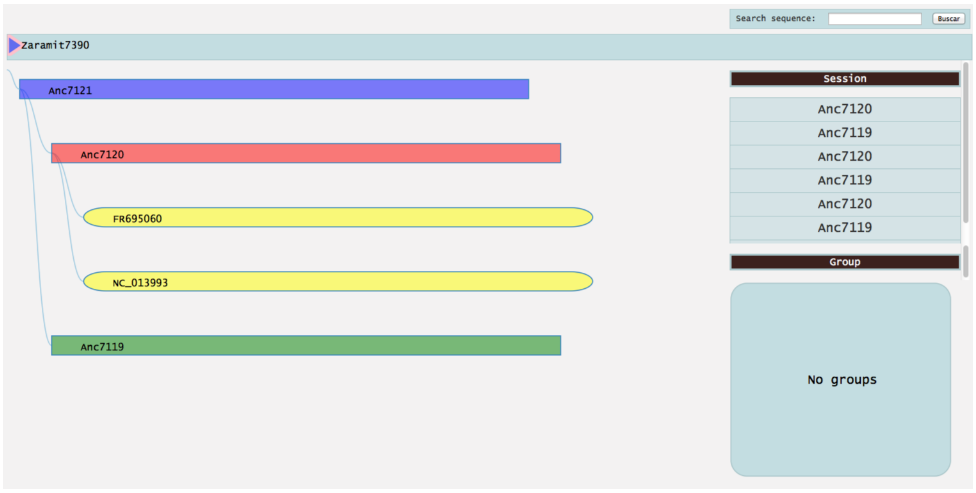


Figure 6.10: Example of how a phylogeny is displayed with PhyloViewer. The rectangles represent internal nodes and the leaves are depicted by rounded rectangles. The search bar is located in the top right side of the image. Below it, there is the navigation history of every node visited during the session of that phylogeny. In addition, at the right bottom part of the image, the group tree is displayed (in this case, no group tree was provided with the phylogeny).

PhyloViewer includes a user-login system that allows the users to keep private and public phylogenies. The private option has been included to support the ongoing work of researchers until the phylogeny is completed. The interface grants a direct edition of the phylogenies too. An example of the phylogenetic storage system of a specific user is displayed in Figure 6.11. PhyloViewer also incorporates an administrative system for the database and user management, as it is shown in Figure 6.12.

A biologist expert in mtDNA endorsed the design and usability of the user interface. Besides, we realized a database load test with 3 synthetic and 3 real phylogenies of different sizes. All these experiments were performed in a desktop computer with an AMD A10-5800k @ 3.80GHz and 8GB of RAM. The time costs of each test is shown in Table 6.1.

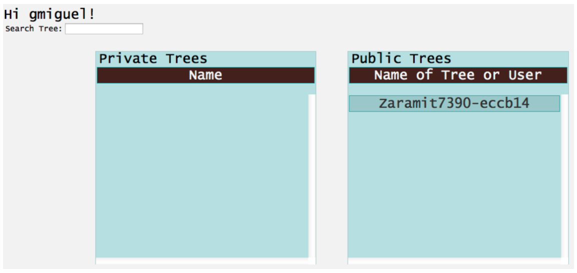


Figure 6.11: Client interface of PhyloViewer where the users can manage their private and public phylogenies.

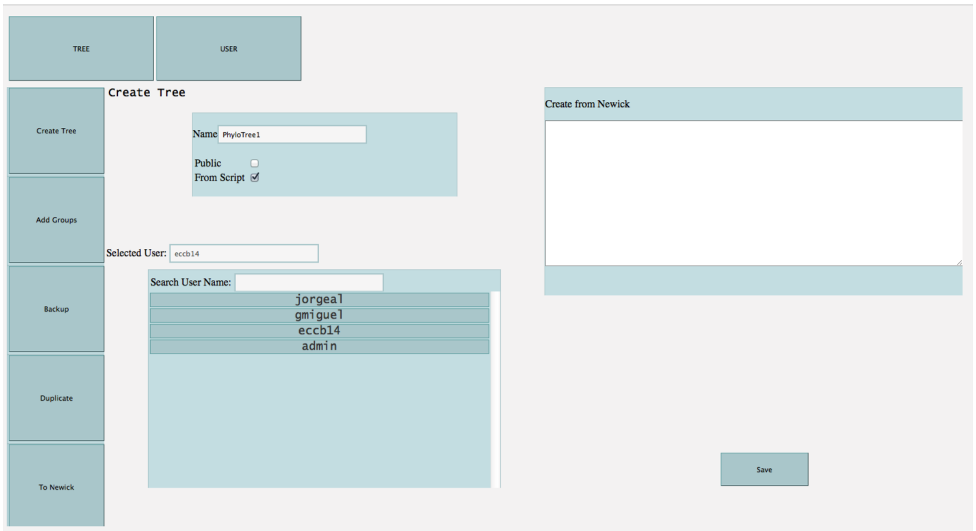


Figure 6.12: Administrative interface of PhyloViewer. It includes the user and phylogeny management, altogether with backup and restore features for the database.

Table 6.1: PhyloViewer's database load time for phylogenies of different type and size.

Tree type	Number of nodes	Size (MB)	Load time (s)
Synthetic	67	0.04	1
	848	0.13	3
	14512	2.01	15
Real	276	0.06	1
	43534	5.11	31
	217666	21.04	130

6.3 Conclusions

In this chapter, we have presented PhyloViewer, a large phylogenetic visualization tool with a parent-children interface and accessible information of each sequence. This new software tool offers a sequence searching, navigation history, the current location within the phylogenetic tree and a window with the group tree corresponding to the current phylogeny. This novel approach solves the usual blur representation of large phylogenies in common visualization tools. Biologists experts on the field of hmtDNA have tested and acknowledged PhyloViewer as a relevant contribution.

As future work, we plan to incorporate to PhyloViewer a multilanguage database management system. The purpose of such improvement is to ease the integration of new bioinformatics scripts and software tools implemented in different programming languages, like Python or Perl. Additionally, we aim to offer an optional interface to those phylogenies that do not have a group tree.

If you torture the data long enough,
it will eventually confess.

Ronald Harry Coase

7 Phylogenetic analysis: *Conservation analysis*

The study of the conservation of each nucleotide at each position of the input biological sequences is a very useful analysis, often connected to phylogenetic information. In this study, protein-coding regions are particularly relevant. Sites where a given nucleotide is present in the majority of the sequences are usually related with essential functionalities of the organism. Thus, a mutation in any of those positions has a high probability to be the cause of one or more diseases. Moreover, the evolutionary distance between the organisms of the input sequences establishes the level of significance and sensitivity of the analysis: the further is their relationship, the more vital is the conserved site. The research and results presented in this chapter are the outcome of the undergraduate project of Francisco Merino-Casallo [179] and they have been published in the international conference paper [180].

In the first section of this chapter, we cover the related work on conservation analysis. Next, we present our contribution to the field with a new software tool designed to perform efficiently with large datasets. Finally, the last section gathers the conclusions for this chapter.

7.1 Related work

The study of the conservation through time of the nucleotides of each site can be very relevant in molecular evolution studies. For instance, the positions with the highest functional importance in hemoglobin are those where the *heme* chemical compound is bounded. These amino acid sites have a remarkable conservation over millions of years of evolutionary history [181]. By contrast, other positions in the protein show a much higher mutation rate. If the degree of functional constraint dictates how conserved a position is, then identifying conserved regions of a protein is tremendously useful [182]. Conservation analysis is one of the most widely used methods for predicting functionally important residues in protein sequences [183]. In the last couple of decades there have been significant scientific advances based on the association of some non-neutral mutations to different types of cancer [184, 185], as well as to other diseases such as Alzheimer's [186] or Parkinson's [187], among others.

There is no scientific agreement on the best conservation analysis method [183], despite the wide number of methods proposed during the last fifty years [4]. This analysis is usually divided in two different stages. The first one estimates the residue frequencies, and the second one uses these results to calculate the conservation score, better known as CI (***c**onservation **i**ndex*). This score is a position-specific value, which represents how likely a mutation is expected to take place in that site. There are different methodologies to cope with both problems. Mathematical estimations or phylogeny-based algorithms are the most extended methods to estimate the residues frequencies. The CI is usually calculated by entropy- or variance-based approaches, or by the composition of substitution matrices, i.e. by applying evolution models. Besides, some methods preprocess the sequences previously to the conservation analysis performing a clustering based on the similarity between residues [182]. The CI is considered a reliable metric for quantifying residue conservation.

Most conservation analysis methods require a MSA as input. Despite the variety of heuristics and gap penalty functions involved in the MSA software tools, there is, to our knowledge, no study on how those variables affect the subsequent conservation analysis.

7.2 Conservation analysis of large datasets

As we have aforementioned, there is a growing concern regarding the negative impact of the increment of available data on the performance of several published algorithms. We could not find any conservation analysis tool able to cope with this issue. Thus, we have developed a new software tool capable of calculate the conservation analysis of very-large datasets through parallelization and divide-and-conquer techniques [180].

The first version of our conservation analysis tool includes the most extended methodologies that perform their calculation in a column-by-column basis. The methods chosen for the residues frequencies estimation are fully compatible with the ones that will be used in the next stage. Moreover, we selected two of the most consistent methods: *weighted* and *unweighted*. The former assigns a specific weight to each sequence based on its similarity with the rest of the aligned set, aiming to compensate for over-representation among multiple aligned sequences. The latter allocates the same weight to every sequence, considering each one equally significant. We selected two well-known techniques for the calculation of the CI: entropy-based and variance-based methods. There are several proposals for entropy-based methods [4], so we chose the one with better qualities in terms of simplicity of the computation process and accuracy. On one hand, the entropy-based method returns a value that will reach its minimal value when all characters at a given site have equal frequencies. On the other hand, the variance-based method maximizes the returned value at the site occupied by an invariant character whose overall frequency is minimal.

The implementation was realized in Python (version 3.4). As we have claimed, one of the major benefits of our tool is the incorporation of parallelism and a divide-and-conquer methodology, making it suitable for large input MSAs. The main idea underneath was to take advantage of the independence assumption among each site of the input MSA, splitting it into fragments of sequences and generating one processing element per portion. The balance of the fragments is calculated as follows: being m the number of characters per sequence of the MSA and t the number of cores of the CPU, the first $t-1$ threads will have a different fragment of the MSA of $\lceil \frac{m}{t} \rceil$ characters, whilst the last thread will get the remaining chunk. Finally, our application generates a report file containing the most valuable information about the conservation analysis performed.

7.2.1 Performance evaluation

From the set of all complete hmtDNA sequences available at GenBank, we created 9 datasets for the performance test. We got 10000 random sequences and aligned them with MAFFT [29] in its default configuration (`--auto`). Afterwards, we generated 9 datasets: 3 of 100 sequences, 3 of 1000 sequences and 3 of 10000 sequences. In turn, for the same number of sequences, each set had a different length: 100, 1000 and 10000 nucleotides. We tested every possible combination of the methods included in our software tool for both the parallelized version and a modified sequential version, removing the thread generation of our program. We also run the tests several times to add statistical significance to our results. All the tests were performed on an Intel(R) Core(TM) i5-4440 CPU @ 3.10GHz with 16G (2x8) DIMM DDR3 1600 MHz. Given the characteristics of this computer, the theoretical speedup is 4, an unreachable boundary because we were only parallelizing frequencies estimation and the conservation score computation. Thus, there were still sections of the algorithm that were executed sequentially.

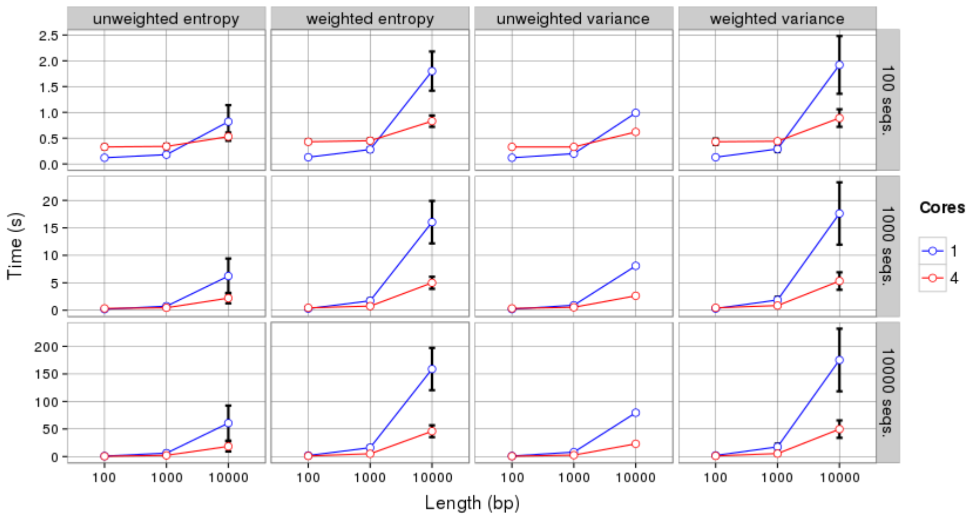


Figure 7.1: Comparison of the average execution time and standard deviation between 100 runs of both sequential and parallel (4 cores) versions of the algorithm. Note that sometimes the standard deviation is so small it is negligible.

As we can see in Figure 7.1, the performance tests carried out proved that the execution time of both approaches was more sensitive to a ten-fold increment of the sequences length than by the same increment in the number of sequences. These performance tests also showed that if the dataset to analyze is rather small (100 sequences and up to 1000 residues or 1000 sequences and up to 100 residues), using the parallel version instead of the sequential one results in an actual degradation of performance. This is due to the additional infrastructure required to manage the parallelism and the small percentage of the total execution time that the estimation of both frequencies and conservation represent in these cases. In contrast, as both the number of sequences and its length grow, the experimental speedup approximates the theoretical one.

7.2.2 MSA influence on conservation analysis

For this preliminary study, we chose a large protein-encoded gene of the mtDNA: the ND2. We randomly selected 400 sequences from GenBank of three different disjoint sets of species: one formed by humans (hmtDNA), another constituted by non-human primates (pmtDNA), and a third one composed by non-primate mammals (mmtDNA). Five MSA configurations where chosen for this experiment. Three for MAFFT: default (`--auto`), accuracy-oriented (`--linsi`) and performance-oriented (`--parttree`). Clustal Omega [45] and MUSCLE [46] were run in their default configuration. We generated their corresponding conservation analyses with our software tool for every method available.

The first thing we noticed is the slight increment on the number of gaps as the species of the datasets are further phylogenetically related. A deeper analysis showed that the different MSA configurations performed different distributions of the gaps in the resultants MSAs. Furthermore, the conservation analyses showed that there were some positions where, regarding the MSA tool chosen, the CI varied. For instance, the mmtDNA set analyzed with unweighted frequencies and entropy-based conservation with a 0.99 threshold showed a difference in the 7th site: MAFFT `--linsi` and Clustal Omega had a CI of 0.9895 whilst with the other alternatives it surpassed the aforementioned threshold. We found a critical situation at position 320 of mmtDNA sequences using weighted frequencies and entropy-based conservation with a 0.99 threshold: the CI of this site was 1.00 with Muscle, 0.9847 with Clustal Omega and 0.8180 with any MAFFT configuration. Moreover, this site appertains to the second nucleotide within a codon, the most likely of the three positions to change the resultant amino acid when a mutation occurs. This result proved that the MSA tool has a remarkable influence when analyzing the conservation of a set of biological sequences and thus, the decision of which one to use is more complex than it may seem at first glance.

7.3 Conclusions

In this chapter, we have presented a new software tool conceived to estimate the conservation index of a given MSA. Our new proposal combines some of the best-published techniques to perform the conservation analysis with the composition of readable and useful reports. Taking into account the Big Data problem, we have implemented parallelization and divide-and-conquer techniques in order to improve the efficiency of the analysis without affecting its accuracy. Besides, we have performed a preliminary study about the impact of the MSA process on the conservation analysis. The inputs were three mtDNA sequence sets, each one composed by organisms of diverse evolutionary distances. The results showed significant differences on the conservation analysis of various MSAs produced from the same dataset. The discrepancies among those MSAs were the outcome of the software tool selected or its parameterization.

As future work, we plan to include new methods in each stage involved in the conservation analysis, specifically those related with phylogeny inference processes. Furthermore, we aim to incorporate more types of reports to make our tool more suitable for not considered scenarios. We are currently working on a deep analysis of every aspect of the MSA tools that can affect the conservation analysis and how they should be handled in the specific case of hmtDNA.

A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.

Alan Mathison Turing

8

Phylogenetic analysis: *Pathogenicity prediction on hmtDNA*

The last phylogenetic analysis covered in this dissertation is the study of the susceptibility of each site on biological sequences to trigger a disease once a mutation has occurred. As we have introduced in the previous chapter, some mutations might produce very severe disorders, but many others will have no relevant effects. Thus, it seems appropriate to provide biologists with both criteria and software tools to determine whether a mutation has an effect in the organism or not. In this specific case, we are going to focus on hmtDNA sequences. The results presented in this chapter are the outcome of our collaboration on the PhD research of Dr. Martín-Navarro [188][1] and they have been published in an international journal [189].

In the first section of this chapter, we cover the state-of-the-art of single-site mutation predictors. Afterwards, we present our proposal of a new classifier to select damaging candidates of the hmtDNA genome with improved performance. Finally, the last section assembles the conclusions for this chapter.

8.1 Related work

Protein-coding genes represent 70% of the hmtDNA. Therefore, the majority of the mutations affect these regions. Many mutations in protein-coding genes in the mtDNA are missense point mutations (single-site mutations) that provoke an amino acid substitution. Some of these mutations will produce very severe disorders, but many others will have no relevant effects. Thus, it is a hard task to differentiate the former from the latter, and diverse criteria have been proposed to achieve this goal [190–192].

The functional characterization of a mtDNA missense mutation is an irreplaceable way to determine its effect and potential pathogenicity. Nonetheless, this process is unreasonable in some situations besides its cost and time consumption. Computational predictors are useful, inexpensive and fast tools for checking novel missense mutations reported in patients with a possible mitochondrial disease, assisting in the selection of mutations for subsequent functional evaluations. Several prediction algorithms are available to classify missense mutations in the neutral or harmful categories [193]. Unfortunately, the overall accuracy of these methods is low, and their predictions may mislead subsequent studies based on them [194–196]. Furthermore, most of these predictors focus uniquely on the nuclear genome. Mutpred [196] and

[1]The results in this chapter have been presented in the PhD dissertation of Dr. Martín-Navarro on September 2016, paying special attention to their biological aspect.

PolyPhen-2 [195] are two well-known and extended predictors for nuclear missense mutations. The former uses HGMD (***H**uman **G**enome **M**utation **D**atabase*) [197] for its training, whilst the latter utilizes the HumDiv dataset. Another example is Provean [198], which uses the Humsavar dataset [199] to adjust the cut-off of damaging mutations. All the aforementioned datasets completely neglect mtDNA missense variants (or most of them). Moreover, they are not aware of the special features of mtDNA-encoded proteins.

8.2 Mitoclass.1

In this section, we present Mitoclass.1 [189], the first missense mutation prediction for hmtDNA sequences. Initially, we focused on the generation of the first mtDNA missense mutations dataset, containing both damaging and neutral ones. The first draft of our dataset was obtained from the MITOMAP website [200] on February 2015. The problem with this dataset is that many missense variants reported meet only a reduced number of the established pathogenicity criteria for mtDNA mutations [190, 201] and their pathogenicity is, therefore, doubtful. This problem is solved if we only consider as damaging missense mutations those already associated with a possible mitochondrial disease and that meet, at least, one of these two criteria: **i)** functional confirmations on published studies; and, **ii)** rareness of the mutation. The latter is based on the presence of the mutation in patients suffering from mitochondriopathy and the virtually absence of it in the control (healthy) population. The final mdmv.1 (***mtD**NA **m**issense **v**ariants.1*) dataset includes 57 damaging and 2778 neutral variants.

The next step was to determine the discriminatory features of our classifier. We have characterized three features: **i)** CI plus cMI (***c**umulative **M**utual **In**formation*) in Eukaryota; **ii)** conservation of the mutant amino acid for each single position in the proteins; and, **iii)** relative frequency for each variant into a particular domain. The first feature is the sum of the two scores.

The CI has been found to be the single most powerful attribute [183, 202]. It has been computed regarding the amino acids present in the rCRS. A high CI for a particular protein position gives an idea about its functional importance. However, a low CI does not directly imply lack of functional importance: there might exist compensations between diverse sites throughout the evolution [203]. Mutual Information from information theory can be used to estimate the extent of the coevolution. The cMI defines to what degree a given amino acid is involved in a mutual information network. This value has been normalized taking as baselines the minimum and maximum cMI values for each protein. A similar consideration about evolutionary conservation and functional importance can be applied for the mutant amino acid. Thus, the second feature computes the CI of the mutant amino acids, which should be very low, particularly in very deleterious positions where natural selection would tend to remove them. The third feature takes into account the location of the protein in the mitochondria to quantify the importance of a specific amino acid substitution. Every hmtDNA protein can be divided into three distinct domains regarding their biochemical environment. Hence, the same amino acid substitutions in different domains will not have the same functional effects [204]. We computed the relative frequency of the variants per domain.

To train the learning algorithm we chose a random sample from the mdmv.1 dataset composed by 60% damaging and 60% neutral variants. The remaining 40% was left to validate the predictor. Due to the imbalanced nature of the training set, where damaging and neutral mutations are not equally represented, we used SMOTE [205] to synthetically oversample the minority class (damaging mutations). We compared the predictive results returned by Random forest, IBK, SMO (***S**equential **M**inimal **O**ptimization*), Naive Bayes Multinomial and SVM (***S**upport **V**ector **M**achine*) [206] classifiers on the validation dataset. The comparison was performed using the default parameters offered by the open-source data mining suite WEKA [207] v3.7.7. Finally, we chose the best one, SMV, as the framework of Mitoclass.1. It was executed with WEKA v3.7.7 and the parameters C and γ were optimized by a grid search using 10-fold cross validation of the training dataset. Finally, $C = 200$ and $\gamma = 0.01$ were selected as the best configuration. The 10-fold cross validation was also executed with the training dataset to check the robustness of the method and prevent the possibility of overfitting.

We have compared our classifier with other very popular predictors, such as PolyPhen-2 (with HumDiv classifier model and setting both “probably damaging” and “possibly damaging” predictions as damaging), Provean (in its default settings), and the results on mtDNA mutations previously reported using Mutpred (score cut-off 0.75) [208]. The first test was performed within these three predictors with the complete mdmv.1 dataset. The results are presented in Table 8.1. We excluded Mitoclass.1 of this first test due to the unreliable results we would get given that its chosen training dataset was part of the mdmv.1 dataset. The analysis shows that PolyPhen-2 is the most accurate predictor with a sensitivity of 94.7% and only 3 FP. Mutpred has the worst results, so we discarded it for the screening of mtDNA missense variants because it would remove too many potential damaging mtDNA mutations.

Table 8.1: Comparison between predictors with the complete mdmv.1 dataset. Sensitivity is estimated as [TP/(TP+FN)], specificity as [TN/(TN+FP)].

	PolyPhen-2	Provean	Mutpred
Sensitivity	94.7%	87.7%	57.9%
Specificity	46.9%	59.2%	87.3%
TP	54	50	33
TN	1303	1646	2426
FP	1475	1132	352
FN	3	7	24

Our second analysis tested the performance of the four predictors for the validation dataset aforementioned extracted from the mdmv.1 dataset. The results are displayed in Table 8.2. Mitcolass.1 achieves the best sensitivity and the best ratio sensitivity/specificity of all of them. Furthermore, our classifier generates results for 100% of the analyzed variants whilst others, like PolyPhen-2, does not generate predictions for ten mutations, classifying them as “unknown”. A further analysis of the FP predictions, we observed that both Provean and PolyPhen-2 classified as neutral a confirmed pathogenic mutation [209]. There are two other separate cases, one for each tool, similar to the previous one. These three mutations affect positions with low conservation in Eukaryota. Nevertheless, our classifier achieves a correct prediction for the great majority of them, because we do not use the conservation of a single position as a discriminatory feature. The only damaging mutation that our classifier does not predict as pathological is one with a very high relative frequency in eukaryotes (*feature 2* = 35%).

Table 8.2: Comparison between predictors with validation dataset of 1100 mutations (23 damaging + 1077 neutral). Sensitivity is estimated as [TP/(TP+FN)], specificity as [TN/(TN+FP)].

	Mitoclass.1	PolyPhen-2	Provean	Mutpred
Sensitivity	95.7%	91.3%	91.3%	60.9%
Specificity	58.7%	47.7%	60.4%	85.6%
TP	22	21	21	14
TN	623	514	650	922
FP	454	563	427	155
FN	1	2	2	9

8.3 Conclusions

In this chapter, we have presented Mitoclass.1, a SVM classifier designed to predict pathogenicity of hmtDNA missense variants. This new software tool is a good screening classifier to select candidate damaging mtDNA missense mutations from patients suffering mitochondrial disorders. Due to the inexistence of well-curated datasets of mtDNA variants, we have developed a new one under pathogenicity criteria called mdmv.1. The chosen discriminatory attributes are based on conservation and coevolution. We also introduce the novel idea of analyzing each protein domain separately. We have trained and validated Mitoclass.1 with our curated dataset of mtDNA amino acid substitutions. The training of our predictor only with previously curated mtDNA variants as well as the selection of discriminatory features improves the performance when compared with other existing predictors. Furthermore, we have provided predictive results with our classifier for all possible missense mutations of the thirteen proteins encoded by hmtDNA.

The number of mtDNA reference sequences available from different species and the number of candidate mutations identified by sequencing is growing very fast. Moreover, the accuracy of Mitoclass.1 relies in its discriminatory features that are dependent of this information. Hence, as future work, we plan to update Mitoclass.1 periodically by retraining it with new data.

The only source of knowledge is experience.

Albert Einstein

9 Conclusiones

Esta tesis doctoral se ha centrado en el desarrollo de metodologías y herramientas software novedosas, prácticas y precisas en el campo de la inferencia y el análisis filogenético molecular. Como elemento adicional, se ha prestado especial atención a la producción de soluciones escalables y eficientes para el procesamiento de grandes volúmenes de datos. Esta decisión ha sido fomentada por la incorporación de la mayoría de temas bioinformáticos a la categoría *Big Data* durante las dos últimas décadas. A lo largo de esta tesis se han expuesto cada una de las fases relacionadas con la inferencia de árboles evolutivos, desde la recopilación de las secuencias biológicas hasta la estimación de la filogenia, tanto de forma individual como a través del uso de sistemas software diseñados para dicho propósito. Además, en los últimos tres capítulos se han recogido análisis filogenéticos que se pueden aplicar sobre los resultados recopilados para su estudio. Todavía existen muchos problemas abiertos, tanto en filogenética como en bioinformática, para los que se van dando soluciones ya sea mediante métodos formales como a través de nuevas aproximaciones empleando eficientemente plataformas de computación. Desde nuestro punto de vista, la clave del éxito de un nuevo algoritmo o herramienta informática reside en que éste proporcione un balance adecuado entre el coste temporal y la solidez estadística de los resultados. En esta tesis se ha presentado el ADN mitocondrial como el caso real de aplicación principal en nuestra investigación. Este tipo de secuencias biológicas tiene una serie de propiedades que lo hacen idóneo para los objetivos planteados: tanto el número de secuencias disponibles como su longitud hacen que su manejo sea computacionalmente intensivo y, en algunos casos, intratable con los métodos convencionales; y tiene una probabilidad mayor de sufrir una mutación en comparación con el ADN nuclear, por lo que hasta los árboles evolutivos de un solo organismo suelen contener información relevante para su estudio. Como nota adicional, las mutaciones que afectan al ADN mitocondrial juegan un papel esencial en varias enfermedades graves, como la miopatía mitocondrial, una enfermedad que afecta al buen funcionamiento de las fibras musculares.

Los principales resultados de esta tesis han sido publicados en congresos internacionales de calidad en bioinformática (destacamos *IEEE BIBM*) y en dos artículos de la revista internacional *BMC Bioinformatics* de alto impacto. A continuación, exponemos las principales características de cada uno de ellos:

- MEvoLib es la primera biblioteca enfocada a estudios sobre evolución molecular implementada en el lenguaje de programación Python. A lo largo de los Capítulos 2, 3 y 4 hemos presentado los distintos módulos que componen la librería, diseñados para facilitar el trabajo tanto de usuarios nóveles como expertos. MEvoLib incluye una recolección automática de secuencias biológicas y árboles filogenéticos tanto de ficheros locales como de bases de datos públicas, como GenBank. Su diseño incluye funcionalidades ya existentes en otras bibliotecas, como la descarga automática de los datos. Como novedad, este módulo permite combinar datos de distintos orígenes y, tras completarse, genera un informe con toda la información que se considera relevante sobre el proceso, como el número total de elementos (ya sean secuencias o filogenias) y la fecha en la que se ha accedido a las distintas fuentes. Dos propiedades a destacar que se han añadido a la descarga automática de datos es la solicitud de la información en lotes y un proceso de actualización de la información descargada, diseñados para optimizar y reducir los tiempos de descarga, respectivamente. MEvoLib también ofrece cuatro métodos para la división de secuencias biológicas en conjuntos de menor dimensión. Los dos más sencillos, dividen las secuencias en conjuntos de secuencias completas o por fragmentos. Estos métodos están puramente orientados al aprovechamiento de las arquitecturas y técnicas de paralelización de procesos. Los otros dos métodos hacen uso del conocimiento biológico disponible de las secuencias para realizar la división por conjuntos. Hemos realizado una nueva implementación del método que aplica la descomposición PRD para mejorar su eficiencia: se ha reemplazado la parte recursiva del algoritmo original por una iterativa, que es paralelizada en tiempo de ejecución. El segundo método realiza una división por genes. Este algoritmo es el primero de su tipo que utiliza únicamente los metadatos disponibles de cada secuencia de entrada como base para producir la descomposición. Se han incorporado técnicas de álgebra de conjuntos y estadística poblacional para generar una solución óptima al problema de asociar los múltiples términos referentes al mismo gen. Finalmente, MEvoLib incorpora tres métodos para la ejecución de distintas herramientas filogenéticas: uno para aplicaciones de alineamiento de múltiples secuencias, otro para herramientas de inferencia filogenética, y otro para métodos de construcción de superárboles y árboles de consenso. Los tres métodos comparten dos características comunes: **i)** la conversión automática del formato de entrada y salida si la herramienta seleccionada no soporta o no genera los formatos especificados por el usuario; y, **ii)** la incorporación de un diccionario de palabras clave con los parámetros más comunes, incluido el comportamiento *por defecto*. Además, si estas palabras clave no recogen la configuración deseada por el usuario, éste puede introducirla como un parámetro más del método.

- Se ha estudiado la aplicación de métodos de aprendizaje para hallar la topología implícita para un conjunto de secuencias dado. En concreto, se ha realizado la implementación del algoritmo publicado por la Dra. Ishteva, que hace uso de tensores de cuarto orden para establecer la relación filogenética entre cuartetos de secuencias hasta inferir el árbol evolutivo completo. La implementación realizada mostró resultados prometedores respecto al coste computacional, pero, por desgracia, la elevada imprecisión de los resultados reveló un problema en el algoritmo que impidió su aplicación en nuestro caso. Nuestra principal conclusión es negativa y, en este contexto, más difícilmente publicable por si sola. Este algoritmo sólo es útil y preciso en los casos en que se conoce a priori la distribución de probabilidad de la entrada.

- PhyloFlow es el primer sistema completamente configurable y automático orientado exclusivamente a la creación y ejecución de flujos de trabajo para inferir y analizar árboles filogenéticos. La facilidad en el diseño de flujos de trabajo eficientes para grandes volúmenes de datos, la adaptabilidad a los requisitos del usuario y la mejora de la precisión en los resultados mediante la incorporación de metodologías basadas en el conocimiento biológico son las tres principales características de nuestro sistema. Además, PhyloFlow incluye la ejecución desatendida (automática) de los sistemas configurados previamente. Nuestro sistema incorpora un proceso interactivo novedoso para el diseño de flujos de trabajo que ofrece tanto la selección de las herramientas software como de sus parámetros. La primera versión del sistema incluye un gran número de los métodos y herramientas más utilizados en los estudios de inferencia filogenética. Adicionalmente, PhyloFlow ofrece consejos para ayudar a los usuarios inexpertos con la configuración de sus flujos de trabajo. Nuestro sistema ha sido diseñado siguiendo una estrategia de divide-y-vencerás y aplicando el modelo de *caja negra* que elimina cualquier incompatibilidad entre herramientas software. Como resultado, PhyloFlow puede producir sistemas de inferencia filogenética eficientes incluso con grandes volúmenes de datos. El análisis de carga de los distintos métodos y procesos disponibles nos ha facilitado incluir una estimación del coste temporal de un flujo de trabajo determinado conociendo, a priori, el tamaño de la entrada. Esta caracterización está sujeta a muchas variables, como el número de secuencias, su longitud y la heterogeneidad de las mismas. No sólo se ha demostrado la potencia de PhyloFlow al reproducir otros sistemas filogenéticos publicados anteriormente, sino que, además, el mecanismo de réplica permite mejorar la precisión del sistema al incorporar la opción de elegir cada aplicación software y sus parámetros.

- Como ya se ha visto en la caracterización del sistema filogenético para el estudio de ADN mitocondrial humano creado mediante PhyloFlow (Sección 5.3.3), el coste tanto económico como temporal para inferir grandes filogenias es muy elevado. Pese a que cada mes cientos o incluso miles de secuencias son secuenciadas e incluidas en las bases de datos públicas, los procesos de inferencia filogenética se realizan, como mucho, una vez al año. Esta situación hace que dichos árboles evolutivos queden en un estado *desactualizado* en unos meses. Además, la validación de estas secuencias se continúa haciendo manualmente en la mayoría de los casos, con el consiguiente consumo de tiempo y los posibles errores humanos cometidos en el proceso. Por estos motivos hemos desarrollado PHYSER, un nuevo algoritmo de detección de errores de secuenciación que, como efecto derivado, actualiza el árbol filogenético en el que se basa para validar las secuencias de entrada. Las pruebas realizadas han demostrado que PHYSER detecta adecuadamente los errores de secuenciación e incluye la secuencia en la filogenia en tan solo unos pocos segundos.

- PhyloViewer es nuestra nueva propuesta para la visualización de grandes árboles filogenéticos. Esta herramienta ofrece una interfaz que muestra únicamente un nodo y sus nodos *hijo*, incluyendo toda la información adicional disponible de cada una de las secuencias localizadas en dichos nodos. PhyloViewer incorpora un buscador de secuencias mediante su identificador, un historial de navegación, la situación del nodo actual respecto a la filogenia y una ventana anexa que muestra la filogenia de grupos correspondiente al árbol filogenético, si se ha proporcionado previamente. Esta forma de visualizar la filogenia soluciona el problema de una representación borrosa en la mayoría de las herramientas publicadas cuando el número de hojas es elevado (mayor a 1000 secuencias para las resoluciones de pantalla actuales). Tanto la facilidad de manejo como la disposición de la información en PhyloViewer han sido probadas y validadas por biólogos expertos en ADN mitocondrial.

- Hemos creado una herramienta para el análisis de la conservación de alineamientos de secuencias biológicas. Esta nueva propuesta combina algunas de las técnicas mejor valoradas para el análisis de la conservación con la generación de informes comprensibles y útiles. Para hacer frente a las grandes cantidades de secuencias biológicas disponibles hoy en día, hemos incluido en nuestra herramienta técnicas de paralelización y estrategias de divide-y-vencerás para mejorar su eficiencia sin afectar a la precisión del análisis. También hemos realizado un estudio preliminar del impacto que tienen las distintas heurísticas existentes de alineamiento de secuencias en el análisis de la conservación. Los resultados mostraron discrepancias significativas en posiciones concretas para el mismo conjunto de secuencias alineadas mediante distintos métodos. Dichas posiciones correspondían a genes que codifican proteínas, por lo que estas diferencias modifican las conclusiones del análisis.

- Mitoclass.1 es un clasificador SVM diseñado para predecir la patogenicidad de mutaciones puntuales en ADN mitocondrial humano.[1] Esta nueva herramienta ofrece un buen predictor de mutaciones puntuales potencialmente dañinas en ADN mitocondrial de pacientes con trastornos mitocondriales. Para poder realizar la clasificación correctamente, se ha creado un nuevo conjunto de datos con criterios patológicos llamado *mdmv.1*, debido a la inexistencia de conjuntos revisados para variantes de ADN mitocondrial. Los atributos discriminatorios seleccionados para nuestro clasificador se han basado en la conservación y la coevolución. Además, se ha incorporado un nuevo discriminador: el análisis del dominio de la proteína al que pertenece la posición a evaluar. Mitoclass.1 se ha entrenado y validado con nuestro conjunto de datos de substituciones de aminoácidos en el ADN mitocondrial. Nuestro clasificador mejora los resultados de otros predictores, ya que ha sido entrenado únicamente con variantes de ADN mitocondrial previamente verificadas y se han seleccionado las características discriminatorias adecuadas. Por otra parte, gracias a Mitoclass.1 hemos podido aportar nuevos resultados predictivos para todas las mutaciones puntuales de los trece genes que codifican proteínas en el ADN mitocondrial humano.

Los códigos fuente y los ficheros de datos tanto de MEvoLib como de PhyloFlow, PHYSER, PhyloViewer y la herramienta de análisis de la conservación están disponibles en la página web del grupo ZARAMIT[2]. MEvoLib

[1]Esta investigación ha sido presentada también, especialmente en su aspecto más biológico, en la tesis doctoral defendida por el Dr. Martín-Navarro en septiembre de 2016.

[2]*http://www.zaramit.org*

incluye un manual que explica detalladamente los módulos que incorpora y varios ejemplos de uso de cada uno de ellos. Los resultados biológicos de la clasificación realizada por Mitoclass.1 pueden consultarse en la publicación [189].

9.1 Trabajo futuro

El principal esfuerzo hacia futuras mejorar en MEvoLib se va a centrar en la incorporación de nuevas herramientas software para todos los módulos que realizan algún procesamiento de datos. Por ejemplo, planeamos incorporar PRANK y KAlign al conjunto de herramientas de alineamiento de secuencias, PhyML para la inferencia filogenética y SuperFine para la construcción de superárboles. En relación a este trabajo, se añadirán nuevos métodos para aquellos procesos comunes en estudios filogenéticos pero que todavía no se han incluido, como Seqboot, perteneciente a PHYLIP, para realizar el remuestreo (*bootstrapping*) de las secuencias en caso de que la herramienta de inferencia filogenética seleccionada no lo incluya. También se evaluará la incorporación de otros métodos menos habituales como aquellos desarrollados para inferir árboles evolutivos sin necesidad de alinear las secuencias previamente. Se pretende mejorar el método de búsqueda y descarga de datos biológicos mediante la incorporación de las modificaciones realizadas manualmente por el usuario a los conjuntos de datos en el informe. Por ejemplo, se añadiría una nueva línea en el historial cuando una secuencia biológica se haya borrado manualmente. Por otro lado, se ha observado que el guardado en memoria de todos los datos recopilados puede generar problemas cuando la información a reunir es extremadamente grande. En el peor caso, podría generarse una excepción por falta de memoria en tiempo de ejecución, perdiendo todos los datos recopilados hasta el momento. Por ello, se ha planeado modificar esta implementación por otra basada en ficheros que permita manejar prácticamente cualquier volumen de información, aunque este cambio vaya a reducir ligeramente su rendimiento. Finalmente, hemos considerado incluir un nuevo método para la división de secuencias de ADN mitocondrial en haplogrupos, así como mejorar en eficiencia todos aquellos módulos y métodos que lo permitan.

También planeamos renovar el soporte multiplataforma de PhyloFlow para diferentes arquitecturas hardware. Así mismo, pretendemos reemplazar la pregunta en la fase de configuración sobre la arquitectura hardware en la que se va a ejecutar el flujo de trabajo que se está diseñando, por un proceso interno que ajuste los parámetros necesarios según la plataforma disponible en tiempo de ejecución. Por último, se va a renovar el código fuente de PhyloFlow para aprovechar todas las ventajas que ofrece MEvoLib. Así, las mejoras que se introduzcan en este último podrán ser utilizadas casi inmediatamente en PhyloFlow.

Otro de nuestros objetivos es mejorar PHYSER incluyendo nuevos criterios de viabilidad biológica para la clasificación de los errores detectados. Estos consistirán en tener en cuenta eventos como las reversiones, es decir, mutaciones que vuelven a dejar el valor que había desaparecido anteriormente en la filogenia, o el análisis de la conservación realizado previamente a las secuencias ya incluidas en el árbol evolutivo. Las mutaciones a valores que ya han aparecido previamente en esa posición o que están altamente conservados tienen una menor probabilidad de estar relacionados con enfermedades o trastornos.

Nos hemos planteado enriquecer PhyloViewer con un sistema de gestión de bases de datos multilenguaje. La finalidad de dicha modificación es facilitar la integración de nuevas herramientas y scripts bioinformáticos implementados en diferentes lenguajes de programación. Adicionalmente, eliminaremos de la interfaz la ventana con el árbol de grupos para aquellas filogenias que no dispongan de dicha información.

También vamos a incluir nuevos métodos en todas las fases del análisis de la conservación en la nueva versión de nuestra herramienta. En particular, vamos a centrar nuestra atención en los métodos basados en árboles filogenéticos. Además, queremos incluir nuevos informes alternativos para hacer nuestra herramienta más útil en aquellos casos que no han sido contemplados anteriormente. Actualmente estamos realizando un estudio en profundidad de los efectos negativos de las diferentes heurísticas para realizar alineamientos en los análisis de la conservación y cómo solventarlos en el caso del ADN mitocondrial humano.

El número de secuencias de referencia disponibles para ADN mitocondrial de diferentes especies y el número de mutaciones candidatas identificadas mediante secuenciación están creciendo rápidamente. Dado que la precisión de Mitoclass.1 se basa en los factores de discriminación que dependen de dicha información, como trabajo futuro planeamos actualizar nuestro clasificador periódicamente.

Con los resultados expuestos en esta tesis pretendemos demostrar que es posible diseñar métodos bioinformáticos que aborden la aparente contradicción que existe en el desarrollo de soluciones de propósito general, y que además obtengan resultados precisos en casos particulares como el del ADN mitocondrial. Además, deseamos incorporar el concepto de *usabilidad* en bioinformática como un elemento central en el diseño de métodos que se adapten tanto a usuarios nóveles como expertos, tengan o no amplios conocimientos informáticos o biológicos.

Bibliography

[1] I. Wagner and H. Musso, "New Naturally Occurring Amino Acids," *Angewandte Chemie International Edition in English*, vol. 22, no. 11, pp. 816–828, 1983.

[2] D. H. Huson, D. C. Richter, C. Rausch, T. Dezulian, M. Franz, and R. Rupp, "Dendroscope: An interactive viewer for large phylogenetic trees," *BMC bioinformatics*, vol. 8, no. 460, pp. 1–6, 2007.

[3] G. E. Jordan and W. H. Piel, "PhyloWidget: web-based visualizations for the tree of life," *Bioinformatics*, vol. 24, no. 14, pp. 1641–1642, 2008.

[4] F. Johansson and H. Toh, "A comparative study of conservation and variation scores," *BMC Bioinformatics*, vol. 11, no. 388, pp. 1–11, 2010.

[5] K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl, "The protein folding problem," *Annual review of biophysics*, vol. 37, pp. 289–316, 2008.

[6] A. Dwevedi, "Involvement of Bioinformatics in Solving Protein Folding Problem," in *Protein Folding*, pp. 39–48, Springer, 2015.

[7] W. Wiechert, "Modeling and simulation: tools for metabolic engineering," *Journal of biotechnology*, vol. 94, no. 1, pp. 37–63, 2002.

[8] W. A. Welch, S. J. Strath, and A. M. Swartz, "Congruent Validity and Reliability of Two Metabolic Systems to Measure Resting Metabolic Rate," *International journal of sports medicine*, vol. 36, no. 5, pp. 414–418, 2015.

[9] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," *Journal of computational biology*, vol. 1, no. 4, pp. 337–348, 1994.

[10] W. H. E. Day, D. S. Johnson, and D. Sankoff, "The Computational Complexity of Inferring Rooted Phylogenies by Parsimony," *Mathematical Biosciences*, vol. 81, no. 1, pp. 33–42, 1986.

[11] S. Roch, "A short proof that phylogenetic tree reconstruction by maximum likelihood is hard," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 3, no. 1, p. 92, 2006.

[12] J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, *et al.*, "The Bioperl toolkit: Perl modules for the life sciences," *Genome research*, vol. 12, no. 10, pp. 1611–1618, 2002.

[13] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, *et al.*, "Biopython: freely available Python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[14] N. Goto, P. Prins, M. Nakao, R. Bonnal, J. Aerts, and T. Katayama, "BioRuby: bioinformatics software for the Ruby programming language," *Bioinformatics*, vol. 26, no. 20, pp. 2617–2619, 2010.

[15] E. R. Mardis, "The impact of next-generation sequencing technology on genetics," *Trends in genetics*, vol. 24, no. 3, pp. 133–141, 2008.

[16] B. Mole, "Science Visualized: The gene sequencing future is here," *Science News*, vol. 185, no. 3, pp. 32–32, 2014.

[17] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers, "GenBank," *Nucleic Acids Research*, vol. 38, pp. 46–51, 2010.

[18] R. E. Giles, H. Blanc, H. M. Cann, and D. C. Wallace, "Maternal inheritance of human mitochondrial DNA," *Proceedings of the National academy of Sciences*, vol. 77, no. 11, pp. 6715–6719, 1980.

[19] W. M. Brown, M. George, and A. C. Wilson, "Rapid evolution of animal mitochondrial DNA," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 76, no. 4, pp. 1967–1971, 1979.

[20] D. R. Foran, "Relative Degradation of Nuclear and Mitochondrial DNA: An Experimental Approach," *Journal of Forensic Sciences*, vol. 51, no. 4, pp. 766–770, 2006.

[21] E. Ruiz-Pesini and D. C. Wallace, "Evidence for adaptive selection acting on the tRNA and rRNA genes of human mitochondrial DNA," *Human Mutation, Variations, Informatics and Disease*, vol. 27, pp. 1072–1081, 2006.

[22] D. C. Wallace, E. Ruiz-Pesini, and D. Mishmar, "mtDNA variation, climatic adaptation, degenerative diseases, and longevity," in *Cold Spring Harbor symposia on quantitative biology*, vol. 68, pp. 471–478, Cold Spring Harbor Laboratory Press, 2003.

[23] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow, "Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees," *Science*, vol. 324, no. 5934, pp. 1561–1564, 2009.

[24] K. Liu, T. J. Warnow, M. T. Holder, S. M. Nelesen, J. Yu, A. P. Stamatakis, and C. R. Linder, "SATe-II: very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees," *Systematic biology*, vol. 61, no. 1, pp. 90–106, 2012.

[25] S. Nelesen, K. Liu, L.-S. Wang, K, C. R. Linder, and T. Warnow, "DACTAL: divide-and-conquer trees (almost) without alignments," *Bioinformatics*, vol. 28, pp. i274–i282, 2012.

[26] R. Blanco and E. Mayordomo, "ZARAMIT: a system for the evolutionary study of human mitochondrial DNA," in *IWANN 2009, Part II*, vol. 5518 of *Lecture Notes in Computer Science*, pp. 1139–1142, 2009.

[27] R. Blanco, E. Mayordomo, J. Montoya, and E. Ruiz-Pesini, "Rebooting the human mitochondrial phylogeny: an automated and scalable methodology with expert knowledge," *BMC Bioinformatics*, vol. 12, no. 174, pp. 1–13, 2011.

[28] J. Álvarez-Jarreta and E. Ruiz-Pesini, "MEvoLib v1.0: the First Molecular Evolution Library for Python," *BMC Bioinformatics*, vol. 17, no. 436, pp. 1–8, 2016.

[29] K. Katoh, K. Misawa, K. Kuma, and T. Miyata, "MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform," *Nucleic Acids Research*, vol. 30, no. 14, pp. 3059–3066, 2002.

[30] C. Burks, "Molecular Biology Database List," *Nucleic Acids Research*, vol. 27, no. 1, pp. 1–9, 1999.

[31] "National Center for Biotechnology Information (NCBI)." *https://www.ncbi.nlm.nih.gov/*.

[32] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[33] A. M. Kogelnik, M. T. Lott, M. D. Brown, S. B. Navathe, and D. C. Wallace, "MITOMAP: a human mitochondrial genome database," *Nucleic acids research*, vol. 24, no. 1, pp. 177–179, 1996.

[34] E. Ruiz-Pesini, M. T. Lott, V. Procaccio, J. C. Poole, M. C. Brandon, D. Mishmar, *et al.*, "An enhanced MITOMAP with a global mtDNA mutational phylogeny," *Nucleic acids research*, vol. 35, no. suppl 1, pp. D823–D828, 2007.

[35] G. D. Schuler, J. A. Epstein, H. Ohkawa, and J. A. Kans, "[10] Entrez: Molecular biology database and retrieval system," *Methods in enzymology*, vol. 266, pp. 141–162, 1996.

[36] R. Leinonen, H. Sugawara, and M. Shumway, "The sequence read archive," *Nucleic acids research*, pp. D19–D21, 2010.

[37] R. M. Schaaper, "Base selection, proofreading, and mismatch repair during DNA replication in Escherichia coli," *Journal of Biological Chemistry*, vol. 268, no. 32, pp. 23762–23765, 1993.

[38] J. M. Berg, J. L. Tymoczko, and L. Stryer, "Biochemistry," 2002. 5th edition.

[39] H. S. Zaher and R. Green, "Fidelity at the molecular level: lessons from protein synthesis," *Cell*, vol. 136, no. 4, pp. 746–762, 2009.

[40] L. Liu, Y. Li, S. Li, N. Hu, Y. He, R. Pong, *et al.*, "Comparison of next-generation sequencing systems," *BioMed Research International*, vol. 2012, no. 251364, pp. 1–11, 2012.

[41] M. van Oven and M. Kayser, "Updated comprehensive phylogenetic tree of global human mitochondrial DNA variation," *Human Mutation*, vol. 30, no. 2, pp. E386–E394, 2009.

[42] H. Carrillo and D. Lipman, "The multiple sequence alignment problem in biology," *SIAM Journal on Applied Mathematics*, vol. 48, no. 5, pp. 1073–1082, 1988.

[43] P. Bonizzoni and G. Della Vedova, "The complexity of multiple sequence alignment with SP-score that is a metric," *Theoretical Computer Science*, vol. 259, no. 1, pp. 63–79, 2001.

[44] W. Just, "Computational complexity of multiple sequence alignment with SP-score," *Journal of computational biology*, vol. 8, no. 6, pp. 615–623, 2001.

[45] F. Sievers, A. Wilm, D. G. Dineen, T. J. Gibson, K. Karplus, W. Li, *et al.*, "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega," *Molecular Systems Biology*, vol. 7, no. 539, pp. 1–6, 2011.

[46] R. C. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Research*, vol. 32, no. 5, pp. 1792–1797, 2004.

[47] T. Lassmann and E. L. Sonnhammer, "Kalign – an accurate and fast multiple sequence alignment algorithm," *BMC bioinformatics*, vol. 6, no. 298, pp. 1–9, 2005.

[48] A. Löytynoja and N. Goldman, "Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis," *Science*, vol. 320, no. 5883, pp. 1632–1635, 2008.

[49] M. Clamp, J. Cuff, S. M. Searle, and G. J. Barton, "The Jalview Java alignment editor," *Bioinformatics*, vol. 20, no. 3, pp. 426–427, 2004.

[50] M. Gouy, S. Guindon, and O. Gascuel, "SeaView version 4: a multiplatform graphical user interface for sequence alignment and phylogenetic tree building," *Molecular biology and evolution*, vol. 27, no. 2, pp. 221–224, 2010.

[51] S. Mirarab, N. Nguyen, S. Guo, L.-S. Wang, J. Kim, and T. Warnow, "PASTA: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences," *Journal of Computational Biology*, vol. 22, no. 5, pp. 377–386, 2015.

[52] U. W. Roshan, T. Warnow, B. M. E. Moret, and T. L. Williams, "Rec-I-DCM3: a fast algorithmic technique for reconstructing phylogenetic trees," in *Computational Systems Bioinformatics Conference, 2004 (CSB 2004)*, pp. 98–109, IEEE, 2004.

[53] D. Navarro-Gomez, J. Leipzig, L. Shen, M. Lott, A. P. M. Stassen, D. C. Wallace, *et al.*, “Phy-Mer: a novel alignment-free and reference-independent mitochondrial haplogroup classifier,” *Bioinformatics*, vol. 31, no. 8, pp. 1310–1312, 2015.

[54] H. Weissensteiner, D. Pacher, A. Kloss-Brandstätter, L. Forer, G. Specht, H. J. Bandelt, *et al.*, “HaploGrep 2: mitochondrial haplogroup classification in the era of high-throughput sequencing,” *Nucleic acids research*, vol. 44, pp. W58–W63, 2016.

[55] R. M. Andrews, I. Kubacka, P. F. Chinnery, R. N. Lightowlers, D. M. Turnbull, and N. Howell, “Reanalysis and revision of the Cambridge reference sequence for human mitochondrial DNA,” *Nature genetics*, vol. 23, no. 2, pp. 147–147, 1999.

[56] C. T. Amemiya, J. Alföldi, A. P. Lee, S. Fan, H. Philippe, I. MacCallum, *et al.*, “The african coelacanth genome provides insights into tetrapod evolution,” *Nature*, vol. 496, no. 7445, pp. 311–316, 2013.

[57] S. Finnilä, M. S. Lehtonen, and K. Majamaa, “Phylogenetic network for European mtDNA,” *The American Journal of Human Genetics*, vol. 68, no. 6, pp. 1475–1484, 2001.

[58] D. Gusfield, S. Eddhu, and C. Langley, “Optimal, efficient reconstruction of phylogenetic networks with constrained recombination,” *Journal of bioinformatics and computational biology*, vol. 2, no. 1, pp. 173–213, 2004.

[59] D. H. Huson and D. Bryant, “Application of phylogenetic networks in evolutionary studies,” *Molecular biology and evolution*, vol. 23, no. 2, pp. 254–267, 2006.

[60] C. Daskalakis and S. Roch, “Alignment-free phylogenetic reconstruction,” in *RECOMB 2010*, vol. 6044 of *Lecture Notes in Computer Science*, pp. 123–137, Springer, Berlin/Heidelberg, 2010.

[61] M. K. Kuhner and J. Felsenstein, “A Simulation Comparison of Phylogeny Algorithms under Equal and Unequal Evolutionary Rates,” *Molecular Biology and Evolution*, vol. 11, no. 3, pp. 459–468, 1994.

[62] Y. Tateno, N. Takezaki, and M. Nei, “Relative efficiencies of the maximum-likelihood, neighbor-joining, and maximum-parsimony methods when substitution rate varies with site,” *Molecular Biology and Evolution*, vol. 11, no. 2, pp. 261–277, 1994.

[63] P. Beerli, "Comparison of Bayesian and maximum-likelihood inference of population genetic parameters," *Bioinformatics*, vol. 22, no. 3, pp. 341–345, 2006.

[64] W. H. E. Day, "Computational complexity of inferring phylogenies from dissimilarity matrices," *Bulletin of mathematical biology*, vol. 49, no. 4, pp. 461–467, 1987.

[65] L. L. Cavalli-Sforza and A. W. F. Edwards, "Phylogenetic analysis. Models and estimation procedures," *American journal of human genetics*, vol. 19, no. 3 Pt 1, pp. 233–257, 1967.

[66] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Molecular Biology and Evolution*, vol. 4, no. 4, pp. 406–425, 1987.

[67] R. V. Eck and M. O. Dayhoff, *Atlas of Protein Sequence and Structure*, pp. 162–168. Silver Spring, Maryland, 1966.

[68] W. M. Fitch, "Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology," *Systematic Zoology*, vol. 20, no. 4, pp. 406–416, 1971.

[69] O. Gascuel, "BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data," *Molecular Biology and Evolution*, vol. 14, pp. 685–695, 1997.

[70] V. Makarenkov, "T-REX: reconstructing and visualizing phylogenetic trees and reticulation networks," *Bioinformatics*, vol. 17, no. 7, pp. 664–668, 2001.

[71] D. L. Swofford, "PAUP*. Phylogenetic analysis using parsimony (* and other methods). Version 4.," 2003.

[72] S. Kumar, K. Tamura, and M. Nei, "MEGA: Molecular Evolutionary Genetics Analysis software for microcomputers," *Computer Applications in the Biosciences*, vol. 10, no. 2, pp. 189–191, 1994.

[73] S. Kumar, G. Stecher, and K. Tamura, "MEGA7: Molecular Evolutionary Genetics Analysis version 7.0 for bigger datasets," *Molecular biology and evolution*, vol. 33, no. 7, pp. 1870—-1874, 2016.

[74] J. Felsenstein, "Evolutionary trees from DNA sequences: A maximum likelihood approach," *Molecular Evolution*, vol. 17, pp. 368–376, 1981.

[75] B. Chor and T. Tuller, “Maximum likelihood of evolutionary trees: hardness and approximation,” *Bioinformatics*, vol. 21, no. 1, pp. 97–106, 2005.

[76] G. J. Olsen, H. Matsuda, R. Hagstrom, and R. Overbeek, “fastDNAml: a tool for construction of phylogenetic trees of DNA sequences using maximum likelihood,” *Computer applications in the biosciences: CABIOS*, vol. 10, no. 1, pp. 41–48, 1994.

[77] S. Guindon and O. Gascuel, “A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood,” *Systematic biology*, vol. 52, no. 5, pp. 696–704, 2003.

[78] S. Guindon, J.-F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel, “New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0,” *Systematic biology*, vol. 59, no. 3, pp. 307–321, 2010.

[79] A. Stamatakis, “RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models,” *Bioinformatics*, vol. 22, no. 21, pp. 2688–2690, 2006.

[80] A. Stamatakis, “RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies,” *Bioinformatics*, vol. 30, no. 9, pp. 1312–1313, 2014.

[81] M. N. Price, P. S. Dehal, and A. P. Arkin, “FastTree: Computing Large Minimum-Evolution Trees with Profiles instead of a Distance Matrix,” *Molecular Biology and Evolution*, vol. 26, pp. 1641–1650, 2009.

[82] M. N. Price, P. S. Dehal, and A. P. Arkin, “FastTree 2–approximately maximum-likelihood trees for large alignments,” *PloS one*, vol. 5, no. 3, p. e9490, 2010.

[83] S. Ota and W. H. Li, “NJML: a hybrid algorithm for the neighbor-joining and maximum-likelihood methods,” *Molecular Biology and Evolution*, vol. 17, no. 9, pp. 1401–1409, 2000.

[84] S. Ota and W. H. Li, “NJML+: an extension of the NJML method to handle protein sequence data and computer software implementation,” *Molecular biology and evolution*, vol. 18, no. 11, pp. 1983–1992, 2001.

[85] T. H. Jukes and C. R. Cantor, *Evolution of protein molecules*, vol. 3, ch. 24, pp. 21–132. New York, 1969.

[86] M. Hasegawa, H. Kishino, and T. Yano, “Dating of the human-ape splitting by a molecular clock of mitochondrial DNA,” *Journal of molecular evolution*, vol. 22, no. 2, pp. 160–174, 1985.

[87] S. Tavaré, “Some probabilistic and statistical problems in the analysis of DNA sequences,” *Lectures on mathematics in the life sciences*, vol. 17, pp. 57–86, 1986.

[88] D. T. Jones, W. R. Taylor, and J. M. Thornton, “The rapid generation of mutation data matrices from protein sequences,” *Computer applications in the biosciences: CABIOS*, vol. 8, no. 3, pp. 275–282, 1992.

[89] S. Whelan and N. Goldman, “A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach,” *Molecular biology and evolution*, vol. 18, no. 5, pp. 691–699, 2001.

[90] K. Liu, C. R. Linder, and T. Warnow, “RAxML and FastTree: Comparing Two Methods for Large-Scale Maximum Likelihood Phylogeny Estimation,” *PLoS One*, vol. 6, no. 11, p. e27731, 2011.

[91] D. Posada and K. A. Crandall, “Modeltest: testing the model of DNA substitution,” *Bioinformatics*, vol. 14, no. 9, pp. 817–818, 1998.

[92] D. Darriba, G. L. Taboada, R. Doallo, and D. Posada, “jModelTest 2: more models, new heuristics and parallel computing,” *Nature methods*, vol. 9, no. 8, p. 772, 2012.

[93] F. Abascal, R. Zardoya, and D. Posada, “Prottest: selection of best-fit models of protein evolution,” *Bioinformatics*, vol. 21, no. 9, pp. 2104–2105, 2005.

[94] D. Darriba, G. L. Taboada, R. Doallo, and D. Posada, “ProtTest 3: fast selection of best-fit models of protein evolution,” *Bioinformatics*, vol. 27, no. 8, pp. 1164–1165, 2011.

[95] Z. Yang and B. Rannala, “Bayesian phylogenetic inference using DNA sequences: a Markov Chain Monte Carlo method,” *Molecular biology and evolution*, vol. 14, no. 7, pp. 717–724, 1997.

[96] J. P. Huelsenbeck and F. Ronquist, “MRBAYES: Bayesian inference of phylogenetic trees,” *Bioinformatics*, vol. 17, no. 8, pp. 754–755, 2001.

[97] F. Ronquist, M. Teslenko, P. van der Mark, D. L. Ayres, A. Darling, S. Höhna, *et al.*, "MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space," *Systematic biology*, vol. 61, no. 3, pp. 539–542, 2012.

[98] M. A. Suchard and B. D. Redelings, "BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny," *Bioinformatics*, vol. 22, pp. 2047–2048, 2006.

[99] J. Felsenstein, "Confidence limits on phylogenies: an approach using the bootstrap," *Evolution*, vol. 39, no. 4, pp. 783–791, 1985.

[100] J. Felsenstein, "Phylogeny inference package (PHYLIP)," 2006. University of Washington, Seattle.

[101] T. Margush and F. R. McMorris, "Consensus*n*-trees," *Bulletin of Mathematical Biology*, vol. 43, no. 2, pp. 239–244, 1981.

[102] G. Nelson, "Cladistic analysis and synthesis: principles and definitions, with a historical note on Adanson's Familles des Plantes (1763–1764)," *Systematic Biology*, vol. 28, no. 1, pp. 1–21, 1979.

[103] E. N. Adams, "Consensus techniques and the comparison of taxonomic trees," *Systematic Biology*, vol. 21, no. 4, pp. 390–397, 1972.

[104] W. F. Day, "Optimal algorithms for comparing trees with labeled leaves," *Journal of classification*, vol. 2, no. 1, pp. 7–28, 1985.

[105] E. N. Adams, "N-trees as nestings: complexity, similarity, and consensus," *Journal of Classification*, vol. 3, no. 2, pp. 299–317, 1986.

[106] R. D. M. Page, "COMMENTS ON COMPONENT-COMPATIBILITY IN HISTORICAL BIOGEOGRAPHY," *Cladistics*, vol. 5, no. 2, pp. 167–182, 1989.

[107] M. J. Sanderson, A. Purvis, and C. Henze, "Phylogenetic supertrees: assembling the trees of life," *Trends in Ecology & Evolution*, vol. 13, no. 3, pp. 105–109, 1998.

[108] B. R. Baum, "Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees," *Taxon*, pp. 3–10, 1992.

[109] M. A. Ragan, “Phylogenetic inference based on matrix representation of trees,” *Molecular phylogenetics and evolution*, vol. 1, no. 1, pp. 53–58, 1992.

[110] M. Steel and A. Rodrigo, “Maximum likelihood supertrees,” *Systematic biology*, vol. 57, no. 2, pp. 243–250, 2008.

[111] F. Ronquist, J. P. Huelsenbeck, and T. Britton, “Bayesian supertrees,” in *Phylogenetic Supertrees*, pp. 193–224, Springer, 2004.

[112] L. R. Foulds and R. L. Graham, “The Steiner problem in phylogeny is NP-complete,” *Advances in Applied Mathematics*, vol. 3, no. 1, pp. 43–49, 1982.

[113] C. Semple and M. Steel, “A supertree method for rooted trees,” *Discrete Applied Mathematics*, vol. 105, no. 1, pp. 147–158, 2000.

[114] M. S. Swenson, R. Suri, C. R. Linder, and T. Warnow, “SuperFine: fast and accurate supertree estimation,” *Systems Biology*, vol. 61, no. 2, pp. 214–227, 2012.

[115] J. P. Huelsenbeck and B. Rannala, “Frequentist properties of Bayesian posterior probabilities of phylogenetic trees under simple and complex substitution models,” *Systematic biology*, vol. 53, no. 6, pp. 904–913, 2004.

[116] D. F. Robinson and L. R. Foulds, “Comparison of phylogenetic trees,” *Mathematical biosciences*, vol. 53, no. 1, pp. 131–147, 1981.

[117] C. Than, D. Ruths, and L. Nakhleh, “PhyloNet: a software package for analyzing and reconstructing reticulate evolutionary relationships,” *BMC bioinformatics*, vol. 9, no. 322, pp. 1–16, 2008.

[118] M. Ishteva, H. Park, and L. Song, “Unfolding Latent Tree Structures using 4th Order Tensors,” in *30th International Conference on Machine Learning*, vol. 3, pp. 316–324, 2013.

[119] C. Semple and M. A. Steel, *Phylogenetics*, vol. 24. Oxford University Press on Demand, 2003.

[120] P. L. Erdös, M. A. Steel, L. A. Székely, and T. J. Warnow, “A few logs suffice to build (almost) all trees: Part II,” *Theoretical Computer Science*, vol. 221, no. 1-2, pp. 77–118, 1999.

[121] A. Anandkumar, K. Chaudhuri, D. J. Hsu, S. M. Kakade, L. Song, and T. Zhang, "Spectral methods for learning multivariate latent tree structure," in *Advances in neural information processing systems*, pp. 2025–2033, 2011.

[122] G. Guennebaud, "Eigen: a C++ linear algebra library." 1st PlaFRIM scientific day, Bordeaux, 2011. *http://eigen.tuxfamily.org/*.

[123] J. Álvarez-Jarreta, G. de Miguel Casado, and E. Mayordomo, "PhyloFlow: A Fully Customizable and Automatic Workflow for Phylogeny Estimation," in *ECCB 2014*, 2014.

[124] J. Álvarez-Jarreta, G. de Miguel Casado, and E. Mayordomo, "PhyloFlow: A Fully Customizable and Automatic Workflow for Phylogenetic Reconstruction," in *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1–7, IEEE, 2014.

[125] J. Álvarez-Jarreta, G. de Miguel Casado, and E. Mayordomo, "Molecular Phylogenetic Analysis: Design and Implementation of Scalable and Reliable Algorithms and Verification of Phylogenetic Properties," in *V Jornadas de Jóvenes Investigadores I3A*, 2016.

[126] J. Álvarez-Jarreta, E. Mayordomo, and E. Ruiz-Pesini, "PHYSER: An Algorithm to Detect Sequencing Errors from Phylogenetic Information," in *6th International Conference on Practical Applications of Computational Biology & Bioinformatics (PACBB 2012)*, pp. 105–112, 2012.

[127] D. L. Swofford, "Phylogenetic analysis using parsimony," 1998.

[128] E. Miguel, "Estudio y análisis de métodos de inferencia filogenética: del ADN a las proteínas," *Undergraduate Project (TAZ-PFC-2012-791), Universidad de Zaragoza*, 2012.

[129] Y. Cao, A. Janke, P. J. Waddell, M. Westerman, O. Takenaka, S. Murata, *et al.*, "Conflict among individual mitochondrial proteins in resolving the phylogeny of eutherian orders," *Journal of Molecular Evolution*, vol. 47, no. 3, pp. 307–322, 1998.

[130] J. Adachi and M. Hasegawa, "Model of amino acid substitution in proteins encoded by mitochondrial DNA," *Journal of molecular evolution*, vol. 42, no. 4, pp. 459–468, 1996.

[131] J. P. Huelsenbeck, P. Joyce, C. Lakner, and F. Ronquist, "Bayesian analysis of amino acid substitution models," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 363, no. 1512, pp. 3941–3953, 2008.

[132] F. Abascal, D. Posada, and R. Zardoya, "MtArt: a new model of amino acid replacement for Arthropoda," *Molecular biology and evolution*, vol. 24, no. 1, pp. 1–5, 2007.

[133] D. A. Bader, U. Roshan, and A. Stamatakis, "Computational Grand Challenges in Assembling the Tree of Life: Problems and Solutions," *Advances in Computers*, vol. 68, pp. 127–176, 2006.

[134] Z. Yang and B. Rannala, "Molecular phylogenetics: principles and practice," *Nature Reviews Genetics*, vol. 13, pp. 303–314, 2012.

[135] D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: From process modeling to workflow automation infrastructure," *Distributed and parallel Databases*, vol. 3, no. 2, pp. 119–153, 1995.

[136] R. Littauer, R. Karthik, B. Ludäscher, W. Michener, and R. Koskela, "Trends in Use of Scientific Workflows: Insights from a Public Repository and Recommendations for Best Practice," *The International Journal of Digital Curation*, vol. 7, no. 2, pp. 92–100, 2012.

[137] A. Luckow, P. Mantha, and S. Jha, "Pilot-Abstraction: A Valid Abstraction for Data-Intensive Applications on HPC, Hadoop and Cloud Infrastructures?," *arXiv preprint arXiv:1501.05041*, 2015.

[138] S. B. Needleman and C. D. Wunsch, "Hardware Accelerators in Computational Biology: Application, Potential, and Challenges," *IEEE Design & Test*, vol. 31, no. 1, pp. 8–18, 2014.

[139] J. Cohen, I. Filippis, M. Woodbridge, D. Bauer, N. Chue Hong, M. Jackson, *et al.*, "RAPPORT: running scientific high-performance computing applications on the cloud," *Philosophical Transactions A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1983, 2012.

[140] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, *et al.*, "Taverna: a tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, 2004.

[141] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, *et al.*, “The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud,” *Nucleic Acids Research*, vol. 41, pp. W557–W561, 2013.

[142] B. Giardine, C. Riemer, R. Hardison, R. Burhans, L. Elnitski, P. Shah, *et al.*, “Galaxy: a platform for interactive large-scale genome analysis,” *Genome Research*, vol. 15, no. 10, pp. 1451–1455, 2005.

[143] M. Abouelhoda, S. Issa, and M. Ghanem, “Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support,” *BMC Bioinformatics*, vol. 13, no. 77, pp. 1–19, 2012.

[144] I. Altintas, J. Wang, D. Crawl, and W. Li, “Challenges and approaches for distributed workflow-driven analysis of large-scale biological data,” in *Workshop on Data analytics in the Cloud at EDBT/ICDT 2012 Conference (DanaC2012)*, pp. 73–78, 2012.

[145] Y. Zhao, Y. Li, I. Raicu, S. Lu, and Z. Xuan, “Architecting Cloud Workflow: Theory and Practice,” in *2014 IEEE International Conference on Computer and Information Technology (CIT)*, pp. 466–473, IEEE Computer Society, 2014.

[146] N. Alachiotis, E. Sotiriades, A. Dollas, and A. Stamatakis, “Exploring FPGAs for accelerating the phylogenetic likelihood function,” in *2009 IEEE International Parallel & Distributed Processing Symposium (IPDPS 2009)*, pp. 1–8, IEEE Computer Society, 2009.

[147] F. Blagojevic, A. Stamatakis, C. D. Antonopoulos, and D. S. Nikolopoulos, “Raxml-cell: Parallel phylogenetic tree inference on the cell broadband engine,” in *2007 IEEE International Parallel & Distributed Processing Symposium (IPDPS 2007)*, pp. 1–10, IEEE Computer Society, 2007.

[148] L. Kuan, J. Neves, F. Pratas, P. Tomás, and L. Sousa, “Accelerating Phylogenetic Inference on GPUs: an OpenACC and CUDA comparison,” in *International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO)*, pp. 589–600, 2014.

[149] T. Majumder, P. P. Pande, and A. Kalyanaraman, “Wireless NoC platforms with dynamic task allocation for maximum likelihood phylogeny reconstruction,” *IEEE Design & Test*, vol. 31, no. 3, pp. 54–64, 2014.

[150] A. M. Kozlov, C. Goll, and A. Stamatakis, “Efficient Computation of the Phylogenetic Likelihood Function on the Intel MIC Architecture,” in *2014 IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPSW 2014)*, pp. 518–527, IEEE Computer Society, 2014.

[151] A. M. Kozlov, A. J. Aberer, and A. Stamatakis, “ExaML version 3: a tool for phylogenomic analyses on supercomputers,” *Bioinformatics*, vol. 1, no. 3, pp. 1–3, 2015.

[152] M. Zakarya, N. Dilawar, and N. Khan, “A Survey on Energy Efficient Load Balancing Algorithms over Multicores,” *International Journal of Research in Computer Applications & Information Technology*, vol. 1, no. 1, pp. 60–68, 2013.

[153] D. Zhao, Z. Zhang, X. Zhou, T. Li, K. Wang, D. Kimpe, *et al.*, “FusionFS: Toward supporting data-intensive scientific applications on extreme-scale high-performance computing systems,” in *2014 IEEE International Conference on Big Data (Big Data)*, pp. 61–70, IEEE Computer Society, 2014.

[154] T. Flouri, F. Izquierdo-Carrasco, D. Darriba, A. J. Aberer, L. T. Nguyen, B. Q. Minh, *et al.*, “The Phylogenetic Likelihood Library,” *Systematic Biology*, vol. 64, no. 2, pp. 356–362, 2015.

[155] S. Jha, J. Qiu, A. Luckow, P. Mantha, and G. C. Fox, “A Tale of Two Data-Intensive Paradigms: Applications, Abstractions, and Architectures,” in *2014 IEEE International Conference on Big Data (Big Data)*, pp. 645–652, IEEE Computer Society, 2014.

[156] M. Margulies, M. Egholm, W. E. Altman, S. Attiya, J. S. Bader, L. A. Bemben, *et al.*, “Genome Sequencing in Open Microfabricated High Density Picoliter Reactors,” *Nature*, vol. 437, pp. 376–380, 2005.

[157] F. A. Matsen, R. B. Kodner, and E. V. Armbrust, “pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree,” *BMC Bioinformatics*, vol. 11, no. 538, pp. 1–16, 2010.

[158] G. Schwarz, “Estimating the dimension of a model,” *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[159] B. L. Maidak, N. Larsen, M. J. McCaughey, R. Overbeek, G. J. Olsen, K. Fogel, *et al.*, "The ribosomal database project," *Nucleic acids research*, vol. 22, no. 17, pp. 3485–3487, 1994.

[160] A. Recuenco, "Formalización y desarollo de un workflow de inferencia filogenética basado en SaaS," *Undergraduate Project (TAZ-TFG-2014-1230), Universidad de Zaragoza*, 2014.

[161] J. Álvarez, R. Blanco, and E. Mayordomo, "Workflows with Model Selection: A Multilocus Approach to Phylogenetic Analysis," in *5th International Conference on Practical Applications of Computational Biology & Bioinformatics (PACBB 2011)*, vol. 93 of *Advances in Intelligent and Soft Computing*, pp. 39–47, Springer Berlin Heidelberg, 2011.

[162] J. Álvarez-Jarreta, G. de Miguel Casado, and E. Mayordomo, "Análisis filogenético molecular: Diseño e implementación de algoritmos escalables y fiables y verificación automática de propiedades de una filogenia," in *I Jornadas de Jóvenes Investigadores I3A*, p. 52, 2012.

[163] B. Ludäscher, M. Weske, T. McPhillips, and S. Bowers, "Scientific Workflows: Business as Usual?," in *Business Process Management* (U. Dayal, J. Eder, J. Koehler, and H. Reijers, eds.), vol. 5701 of *Lecture Notes in Computer Science*, pp. 31–47, Springer Berlin Heidelberg, 2009.

[164] J. Basney, M. Livny, and T. Tannenbaum, "High Throughput Computing with Condor," *HPCU news*, vol. 1, no. 2, 1997.

[165] P. Couvares, T. Kosar, A. Roy, J. Weber, and K. Wenger, "Workflow in Condor," in *In Workflows for e-Science*, Springer Press, 2007.

[166] U. Purkhold, M. Wagner, G. Timmermann, A. Pommerening-Röser, and H. P. Koops, "16S rRNA and amoA-based phylogeny of 12 novel betaproteobacterial ammonia-oxidizing isolates: extension of the dataset and proposal of a new lineage within the nitrosomonads," *International Journal of Systematic and Evolutionary Microbiology*, vol. 53, pp. 1485–1494, 2003.

[167] A. Achilli, C. Rengo, C. Magri, V. Battaglia, A. Olivieri, R. Scozzari, *et al.*, "The molecular dissection of mtDNA haplogroup H confirms that the Franco-Cantabrian glacial refuge was a major source for the

European gene pool," *American Journal of Human Genetics*, vol. 75, pp. 910–918, 2004.

[168] R. Rajkumar, J. Banerjee, H. B. Gunturi, R. Trivedi, and V. K. Kashyap, "Phylogeny and antiquity of M macrohaplogroup inferred from complete mtDNA sequence of Indian specific lineages," *BMC Evolutionary Biology*, vol. 5, no. 26, pp. 1–8, 2005.

[169] S. Guajardo, "Entorno de visualización y edición de árboles para filogenias extensas," *Undergraduate Project (TAZ-PFC-2014-305), Universidad de Zaragoza*, 2014.

[170] J. Álvarez-Jarreta and G. de Miguel Casado, "PhyloViewer: A Phylogenetic Tree Viewer for Extense Phylogenies," in *ECCB 2014*, 2014.

[171] J. Müller and K. Müller, "TreeGraph: automated drawing of complex tree figures using an extensible tree description format," *Molecular Ecology Notes*, vol. 4, no. 4, pp. 786–788, 2004.

[172] A. Rambaut, "FigTree, a graphical viewer of phylogenetic trees," 2007. *http://tree.bio.ed.ac.uk/software/figtree*.

[173] I. Letunic and P. Bork, "Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation," *Bioinformatics*, vol. 23, no. 1, pp. 127–128, 2007.

[174] H. Zhang, S. Gao, M. J. Lercher, S. Hu, and W.-H. Chen, "EvolView, an online tool for visualizing, annotating and managing phylogenetic trees," *Nucleic acids research*, vol. 40, no. W1, pp. W569–W572, 2012.

[175] B. C. Stöver and K. F. Müller, "TreeGraph 2: combining and visualizing evidence from different phylogenetic analyses," *BMC bioinformatics*, vol. 11, no. 7, pp. 1–9, 2010.

[176] "Matplotlib." *http://matplotlib.org/*.

[177] "NetworkX." *http://networkx.github.io/*.

[178] "PyGraphviz." *https://pygraphviz.github.io/*.

[179] F. Merino-Casallo, "Diseño y Estudio de herramientas para el Análisis del Índice de Conservación del ADN mitocondrial," *Undergraduate Project (TAZ-PFC-2014-295), Universidad de Zaragoza*, 2014.

[180] F. Merino-Casallo, J. Álvarez-Jarreta, and E. Mayordomo, "Conservation in mitochondrial DNA: Parallelized estimation and alignment influence," in *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2015)*, pp. 1434–1440, IEEE, 2015.

[181] R. D. M. Page and E. C. Holmes, *Molecular evolution: a phylogenetic approach.* Blackwell Publishing Ltd., 2nd. edition ed., 1998.

[182] W. S. J. Valdar, "Scoring residue conservation," *Proteins: Structure, Function, and Bioinformatics*, vol. 48, no. 2, pp. 227–241, 2002.

[183] J. A. Capra and M. Singh, "Predicting functionally important residues from sequence conservation," *Bioinformatics*, vol. 23, no. 15, pp. 1875–1882, 2007.

[184] J. Carew and P. Huang, "Mitochondrial defects in cancer," *Molecular Cancer*, vol. 1, no. 9, pp. 1–12, 2002.

[185] M. Brandon, P. Baldi, and D. C. Wallace, "Mitochondrial mutations in cancer," *Oncogene*, vol. 25, no. 34, pp. 4647–4662, 2006.

[186] K. Hirai, G. Aliev, A. Nunomura, H. Fujioka, R. L. Russell, C. S. Atwood, *et al.*, "Mitochondrial Abnormalities in Alzheimer's Disease," *The Journal of Neuroscience*, vol. 21, no. 9, pp. 3017–3023, 2001.

[187] A. Bender, K. J. Krishnan, C. M. Morris, G. A. Taylor, A. K. Reeve, R. H. Perry, *et al.*, "High levels of mitochondrial DNA deletions in substantia nigra neurons in aging and Parkinson disease," *Nature Genetics*, vol. 38, no. 5, pp. 515–517, 2006.

[188] A. Martín-Navarro, *MITOCLASS.1, un predictor de patogenicidad para mutaciones no sinónimas en los polipéptidos codificados por el mtDNA humano.* PhD thesis, Universidad de Zaragoza, 2016.

[189] A. Martín-Navarro, A. Gaudioso-Simón, J. Álvarez-Jarreta, J. Montoya, E. Mayordomo, and E. Ruiz-Pesini, "Machine learning classifier for identification of damaging missense mutations exclusive to human mitochondrial DNA-encoded polypeptides," *BMC Bioinformatics*, vol. 18, no. 158, pp. 1–11, 2017.

[190] J. Montoya, E. López-Gallardo, C. Díez-Sánchez, M. J. López-Pérez, and E. Ruiz-Pesini, "20 years of human mtDNA pathologic point mutations: carefully reading the pathogenicity criteria," *Biochimica et*

Biophysica Acta (BBA)-Bioenergetics, vol. 1787, no. 5, pp. 476–483, 2009.

[191] S. DiMauro and E. A. Schon, "Mitochondrial DNA mutations in human disease," *American journal of medical genetics*, vol. 106, no. 1, pp. 18–26, 2001.

[192] J. L. Elson, M. G. Sweeney, V. Procaccio, J. W. Yarham, A. Salas, Q.-P. Kong, *et al.*, "Toward a mtDNA locus-specific mutation database using the LOVD platform," *Human mutation*, vol. 33, no. 9, pp. 1352–1358, 2012.

[193] J. Thusberg and M. Vihinen, "Pathogenic or not? And if so, then how? Studying the effects of missense mutations using bioinformatics methods," *Human mutation*, vol. 30, no. 5, pp. 703–714, 2009.

[194] S. Castellana and T. Mazza, "Congruency in the prediction of pathogenic missense mutations: state-of-the-art web-based tools," *Briefings in bioinformatics*, vol. 14, no. 4, pp. 448–459, 2013.

[195] I. A. Adzhubei, S. Schmidt, L. Peshkin, V. E. Ramensky, A. Gerasimova, P. Bork, *et al.*, "A method and server for predicting damaging missense mutations," *Nature methods*, vol. 7, no. 4, pp. 248–249, 2010.

[196] B. Li, V. G. Krishnan, M. E. Mort, F. Xin, K. K. Kamati, D. N. Cooper, *et al.*, "Automated inference of molecular mechanisms of disease from amino acid substitutions," *Bioinformatics*, vol. 25, no. 21, pp. 2744–2750, 2009.

[197] P. D. Stenson, M. Mort, E. V. Ball, K. Shaw, A. D. Phillips, and D. N. Cooper, "The Human Gene Mutation Database: building a comprehensive mutation repository for clinical and molecular genetics, diagnostic testing and personalized genomic medicine," *Human genetics*, vol. 133, no. 1, pp. 1–9, 2014.

[198] Y. Choi and A. P. Chan, "PROVEAN web server: a tool to predict the functional effect of amino acid substitutions and indels," *Bioinformatics*, vol. 31, no. 16, pp. 2745–2747, 2015.

[199] C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, *et al.*, "The universal protein resource (uniprot): an expanding universe of protein information," *Nucleic acids research*, vol. 34, no. suppl 1, pp. D187–D191, 2006.

[200] "MITOMAP website." *http://www.mitomap.org/MITOMAP*.

[201] L.-J. C. Wong, "Pathogenic mitochondrial DNA mutations in protein-coding genes," *Muscle & nerve*, vol. 36, no. 3, pp. 279–293, 2007.

[202] N. V. Petrova and C. H. Wu, "Prediction of catalytic residues using Support Vector Machine with selected protein sequence and structural properties," *BMC bioinformatics*, vol. 7, no. 312, pp. 1–12, 2006.

[203] S. Castellana, S. Vicario, and C. Saccone, "Evolutionary patterns of the mitochondrial genome in Metazoa: exploring the role of mutation and selection in mitochondrial protein–coding Genes," *Genome biology and evolution*, vol. 3, pp. 1067–1079, 2011.

[204] M. J. Betts and R. B. Russell, *Amino acid properties and consequences of substitutions*, ch. 14, pp. 289–316. Wiley New York, 2003.

[205] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[206] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[207] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[208] L. Pereira, P. Soares, P. Radivojac, B. Li, and D. C. Samuels, "Comparing phylogeny and the predicted pathogenicity of protein variations reveals equal purifying selection across the global human mtDNA diversity," *The American Journal of Human Genetics*, vol. 88, no. 4, pp. 433–439, 2011.

[209] A. L. Mitchell, J. L. Elson, N. Howell, R. W. Taylor, and D. M. Turnbull, "Sequence variation in mitochondrial complex I genes: mutation or polymorphism?," *Journal of medical genetics*, vol. 43, no. 2, pp. 175–179, 2006.

[210] M. V. Han and C. M. Zmasek, "phyloXML: XML for evolutionary biology and comparative genomics," *BMC bioinformatics*, vol. 10, no. 356, pp. 1–6, 2009.

[211] Y.-G. Yao, Q.-P. Kong, A. Salas, and H.-J. Bandelt, “Pseudomitochondrial genome haunts disease studies,” *Journal of medical genetics*, vol. 45, no. 12, pp. 769–772, 2008.

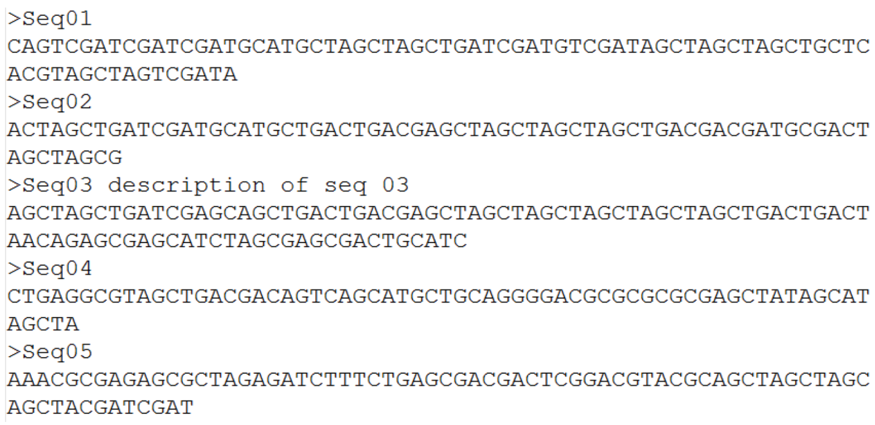
A
Bioinformatics usual file formats

Throughout this dissertation, we have mentioned several usual file formats in bioinformatics. We considered their contents should be explained together with an example. In this appendix, we cover the most frequent text-based formats among the vast number of alternatives available. First, we will describe the formats relating to biological sequences, and, after that, those pertinent to the storage of phylogenetic trees.

A.1 Biological sequence file formats

FASTA

The first format we are going to explain is FASTA. It is the most common format and it can store aligned or unaligned sequences of any type. The file content is a series of sequence blocks with the same structure. The first line contains the greater-than symbol (">") followed by the sequence's identifier. This identifier can be followed by a white space (" ") and a description of the sequence, but this is not required. In the next lines, the sequence will be displayed with a fixed length, usually between 60 and 80 characters. The last line is the only one allowed to ignore this condition. An example of 5 unaligned sequences is shown in Figure A.1.



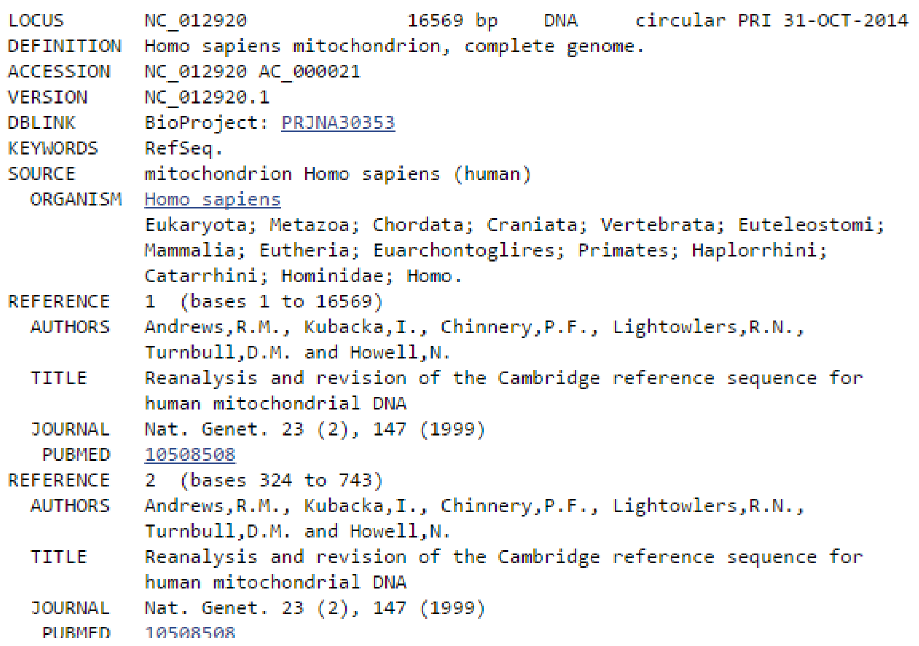
```
>Seq01
CAGTCGATCGATCGATGCATGCTAGCTAGCTGATCGATGTCGATAGCTAGCTAGCTGCTC
ACGTAGCTAGTCGATA
>Seq02
ACTAGCTGATCGATGCATGCTGACTGACGAGCTAGCTAGCTAGCTGACGACGATGCGACT
AGCTAGCG
>Seq03 description of seq 03
AGCTAGCTGATCGAGCAGCTGACTGACGAGCTAGCTAGCTAGCTAGCTAGCTGACTGACT
AACAGAGCGAGCATCTAGCGAGCGACTGCATC
>Seq04
CTGAGGCGTAGCTGACGACAGTCAGCATGCTGCAGGGGACGCGCGCGCGAGCTATAGCAT
AGCTA
>Seq05
AAACGCGAGAGCGCTAGAGATCTTTCTGAGCGACGACTCGGACGTACGCAGCTAGCTAGC
AGCTACGATCGAT
```

Figure A.1: Example of FASTA file format with 5 unaligned DNA sequences.

GENBANK

The GENBANK format was partially covered in Section 3.3.3. It can contain aligned or unaligned sequences of any type, just like FASTA. Each sequence block in these files has three separated sections: **i)** the information section; **ii)** the metadata or features section; and, **iii)** the sequence. The first one is displayed in Figure A.2. It contains relevant information about the sequence's origin, like its identifier, description and the papers in which it has been published. The metadata part has been already shown in Figure 3.1. It mainly encloses the data of each gene or relevant fragment available in the sequence.

The third part contains the sequence itself, showing in each line the index of the first character displayed, followed by up to 60 characters of the



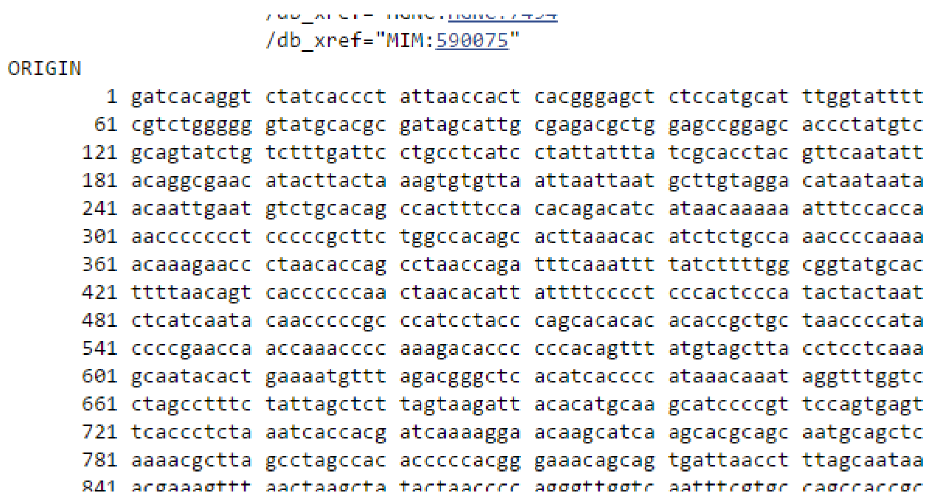
```
LOCUS       NC_012920              16569 bp    DNA     circular PRI 31-OCT-2014
DEFINITION  Homo sapiens mitochondrion, complete genome.
ACCESSION   NC_012920 AC_000021
VERSION     NC_012920.1
DBLINK      BioProject: PRJNA30353
KEYWORDS    RefSeq.
SOURCE      mitochondrion Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
            Catarrhini; Hominidae; Homo.
REFERENCE   1  (bases 1 to 16569)
  AUTHORS   Andrews,R.M., Kubacka,I., Chinnery,P.F., Lightowlers,R.N.,
            Turnbull,D.M. and Howell,N.
  TITLE     Reanalysis and revision of the Cambridge reference sequence for
            human mitochondrial DNA
  JOURNAL   Nat. Genet. 23 (2), 147 (1999)
   PUBMED   10508508
REFERENCE   2  (bases 324 to 743)
  AUTHORS   Andrews,R.M., Kubacka,I., Chinnery,P.F., Lightowlers,R.N.,
            Turnbull,D.M. and Howell,N.
  TITLE     Reanalysis and revision of the Cambridge reference sequence for
            human mitochondrial DNA
  JOURNAL   Nat. Genet. 23 (2), 147 (1999)
   PUBMED   10508508
```

Figure A.2: Example of the information section in a GENBANK file format.

sequence in 10-length sets separated by a blank character (“ ”). An example of this section is shown in Figure A.3. The end of the block is marked by a new line containing only two slashes (“//”) and the new sequence block (if any) will start in the next line.

PHYLIP

The PHYLIP format can store only aligned sequences of any type. The way this format file displays the sequences is quite different from the other two presented. The first line is usually referred to as the *header*, and it contains the dimensions of the alignment separated by one of more spaces. The first positive integer specifies the number of sequences (n) and the second one their length (m). The smallest supported alignment dimensions are 1×1. In the next line the alignment section starts, were we will find n lines, one for each sequence of the alignment. Each line starts with the identifier of the



```
                     /db_xref="MIM:590075"
ORIGIN
        1 gatcacaggt ctatcaccct attaaccact cacgggagct ctccatgcat ttggtatttt
       61 cgtctggggg gtatgcacgc gatagcattg cgagacgctg gagccggagc accctatgtc
      121 gcagtatctg tctttgattc ctgcctcatc ctattattta tcgcacctac gttcaatatt
      181 acaggcgaac atacttacta aagtgtgtta attaattaat gcttgtagga cataataata
      241 acaattgaat gtctgcacag ccactttcca cacagacatc ataacaaaaa atttccacca
      301 aacccccct ccccgcttc tggccacagc acttaaacac atctctgcca aaccccaaaa
      361 acaaagaacc ctaacaccag cctaaccaga tttcaaattt tatcttttgg cggtatgcac
      421 ttttaacagt caccccccaa ctaacacatt attttcccct cccactccca tactactaat
      481 ctcatcaata caaccccgc ccatcctacc cagcacacac acaccgctgc taaccccata
      541 ccccgaacca accaaacccc aaagacaccc cccacagttt atgtagctta cctcctcaaa
      601 gcaatacact gaaaatgttt agacgggctc acatcacccc ataaacaaat aggtttggtc
      661 ctagcctttc tattagctct tagtaagatt acacatgcaa gcatccccgt tccagtgagt
      721 tcaccctcta aatcaccacg atcaaaagga acaagcatca agcacgcagc aatgcagctc
      781 aaaacgctta gcctagccac acccccacgg gaaacagcag tgattaacct ttagcaataa
```

Figure A.3: Example of the sequence section in a GENBANK file format.

sequence followed immediately by up to 60 characters of the sequence. They can be represented in a single string or in chunks of 10 characters separated by a white space (“ ”). The *strict* PHYLIP format forces the identifier to have at most 10 characters, whilst other software tools may relax this restriction. This means the sequence will start in the 11th position in the strict version. If the length of the sequences is longer than 60 characters, we will find an empty line at the end of the first group and the next block of n lines starting at the 61th position of each sequence. The same sequence order as in the previous block is kept, but each sequence will start at the beginning of the line without the identifier. A PHYLIP format example of 5 aligned sequences is displayed in Figure A.4.

The example is usually referred to as interleaved PHYLIP format. A sequential version can also be found where each sequence is displayed completely in the same format. The identifier at the beginning of a line will indicate the starting of a new sequence.

```
 5  68
Seq06     CAGTCGATCG ATCGATGCAT GCTAGCTAGC TGATCGATGT CGA-AGCTAG
Seq07     ACT-GCTGAT CGATGCATGC TGA-TGACGA GCTAGCTAGC TAGCTGACGA
Seq08     AGCTAGCTGA TCGAGCAGCT GACTG-CGAG CTAGCTAGCT AGCTAGCTAG
Seq09     CTGAGGCGTA GCTGACGACA GTCAGCATGC TGC-GGGGAC GCGCGCG-GA
Seq10     AAACGCGAGA GC-CTAGAGA TCTTTCTGAG CGACGACT-G GACGTACGCA

CTAGCTGCTC ACGTAGCT
CGA--CGACT AGCTAGCG
CTGACTGACT AACAGAGC
GCTATAGCAT AGCT--AC
GCTAGCTAGC AGCTACGA
```

Figure A.4: Example of PHYLIP file format with an alignment of 5 DNA sequences.

A.2 Phylogenetic tree file formats

We will now cover two file formats that are designed to store phylogenetic trees. All the examples shown are the text representation of the tree displayed in FigureA.5.

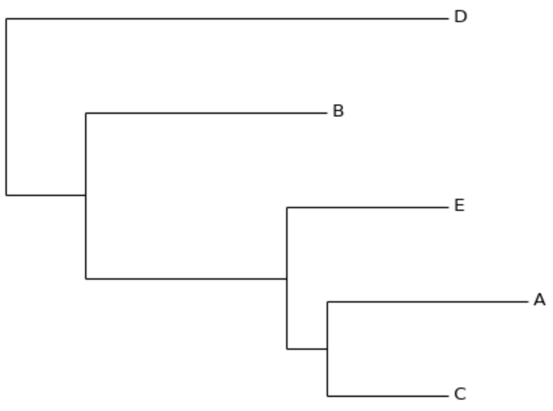


Figure A.5: Phylogeny used as an example to be represented by the tree file formats.

NEWICK

The NEWICK tree format is a way of representing phylogenies with edge lengths using parentheses and commas. This format can describe both rooted and unrooted trees, but, in the latter, an arbitrary node will be chosen as its root for representation purposes. Each node is separated by a coma, and each subtree is delimited by parentheses. The root node of each subtree is displayed at the closure of its corresponding parentheses. The branch length is annotated after each node with a colon (":") followed by a float number. More than one tree can be stored in the same NEWICK file. The end of each tree will be marked by a semicolon (";"), even if only one tree is stored in the file. The following NEWICK format string is equivalent to the phylogenetic tree represented in Figure A.5:

((B:6.0,((A:5.0,C:3.0):1.0,E:4.0):5.0):2.0,D:11.0);

There are other ways of representing the same tree in this format, but the one displayed is the most common one. The alternatives may usually include the inner nodes' names and the root name. The only mandatory symbols are the parentheses, the commas and the semicolon.

PhyloXML

PhyloXML [210] is an XML language for the storage of phylogenies and their associated information. The structure of PhyloXML is described by XSD (*XML **S**chema **D**efinition*) language. The basic description of a phylogeny includes *clades* (subtrees), node names and distances, but there are many more elements to include much more information about the nodes and the tree itself. Thus, phyloXML is a more enriched alternative for phylogenetic representation than the NEWICK format. The text representation of the phylogeny displayed in Figure A.5 in phyloXML format is shown in Figure A.6.

```
<phyloxml xmlns="http://www.phyloxml.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.phyloxml.org http://www.phyloxml.org/1.10/phyloxml.xsd">
  <phylogeny rooted="true">
    <clade>
      <clade>
        <branch_length>2.0</branch_length>
        <clade>
          <name>B</name>
          <branch_length>6.0</branch_length>
        </clade>
        <clade>
          <branch_length>5.0</branch_length>
          <clade>
            <branch_length>1.0</branch_length>
            <clade>
              <name>A</name>
              <branch_length>5.0</branch_length>
            </clade>
            <clade>
              <name>C</name>
              <branch_length>3.0</branch_length>
            </clade>
          </clade>
          <clade>
            <name>E</name>
            <branch_length>4.0</branch_length>
          </clade>
        </clade>
      </clade>
      <clade>
        <name>D</name>
        <branch_length>11.0</branch_length>
      </clade>
    </clade>
  </phylogeny>
</phyloxml>
```

Figure A.6: Example of phyloXML tree format for the phylogeny displayed in Figure A.5.

B
Issues with GenBank's information regarding hmtDNA sequences

During our analysis of a dataset of 31835 complete hmtDNA sequences, we found several issues regarding GenBank's metadata. mtDNA sequences are very frequently obtained in population studies from many individuals without paying enough attention to the quality of individual sequences [211]. Some sequences included the two sections of the D-loop in their raw sequence but they were not tagged in their metadata (e.g. *KR025151.1* and *EU095251.1*). A similar scenario happened with another subset of sequences containing again both fragments but tagging only one of them (e.g. *KJ186009.1* and *GQ214521.3*). There were also sequences whose D-loop boundaries seemed incorrect (e.g. *HM156696.1* and *KR902536.1*). We consider this a serious issue that must be tackled and solved as soon as possible in order to provide the scientific community valuable and accurate data upon which to base their research. Meanwhile, we advise researchers to be particularly careful and pay special attention to this matter.

All the sequences presented in the following tables were downloaded from GenBank on 04/Aug/2016 by querying: *"homo sapiens"[porgn] AND mitochondrion[Filter] NOT mRNA[Filter] AND "complete genome"[All Fields]*. The version of the sequences is not included but all the accessions (identifiers) refer to the latest version available on that date.

Table B.1: Sequences from our initial dataset that did not include a labeled D-loop.

Sequences Identifiers
AB055387, AJ842744 *to* AJ842751, AP008249 *to* AP008920, AP009419 *to* AP009475, AP010661 *to* AP010772, AP010970 *to* AP011059, AP012350 *to* AP012353, AP012365, AY245555, AY665667, DQ112686 *to* DQ112962, X93334, DQ358973 *to* DQ358977, DQ473537, DQ862536 *to* DQ862537, EF177405 *to* EF177447, EU095194 *to* EU095236, EU095238 *to* EU095251, EU935433 *to* EU935467, FJ460520 *to* FJ460562, FJ625845 *to* FJ625860, GQ369957, GU455415 *to* GU455422, KR025151, KT891989 *to* KT891990,

Table B.2: Sequences from our initial dataset that included the D-loop without tagging one or both sections.

Sequences Identifiers
D38112, FJ194437, FN600416, FN673705, FR695060, GQ214520 *to* GQ214527, HQ113226, KF146236 *to* KF146293, KJ185394 *to* KJ186009, KJ533544 *to* KJ533545, KJ871653 *to* KJ871654, KJ882427 *to* KJ882428, KJ882848, KJ890387 *to* KJ890390, KP698374, KP702293

Table B.3: Sequences from our initial dataset that included the D-loop with unreasonable boundaries.

Sequences Identifiers
AY195745 *to* AY195792, AY950289, DQ246830, DQ246833, DQ826448, EF079873 *to* EF079876, EF660929, EU086510, EU089746 *to* EU089747, EU092658 *to* EU092966, EU151466, EU200235, EU200237, EU200347, EU215455, EU215517, EU219920 *to* EU219921, EU232008, EU233277 *to* EU233278, EU294321 *to* EU294323, EU443443 *to* EU443514, EU482319, EU664585 *to* EU664586, EU669888 *to* EU669889, EU721733 *to* EU721734, EU742151, EU753433, EU760854, EU768844, EU770202, EU770310, EU825946, EU828774, EU931680, EU935845, FJ157838 *to* FJ157849, FJ656214, FJ656216 *to* FJ656217, FJ769771, FJ775667, FJ788098, FJ801039, FJ821289, FJ825753, GU002155, GU004258 *to* GU004259, GU012633, GU012637, GU045487, GU048747, GU056815, GU390312 *to* GU390313, GU391321, GU433215, HM047061, HM050402, HM054058, HM055613, HM057816, HM060309, HM156672 *to* HM156696, HQ287872 *to* HQ287898, HQ917079, JF261632, JF262142, JF265069, JF265240, JF275845, JF286633 *to* JF286634, JF747026, JF979198 *to* JF979211, JN030346, JN032298, JN032303, JN034044, JN034636, JN035224, JN035288, JN037468 *to* JN037470, JN038392 *to* JN038393, JN043363, JN048471, JN106183, JN106403, JN107640, JN107813, JN112339, JN212576, JN214391 *to* JN214438, JN214440 *to* JN214444, JN214446 *to* JN214454, JN214457 *to* JN214480, JN224991, JN232198, JN247624, JN252308, JN648827, JN651417, JN657206, JN660158, JN663354, JN663830, JN819535, JN828512, JN828961, JN834028, JN989561, JX414172, JX415317 *to* JX415318, JX417188, JX423390, JX424821, JX462680 *to* JX462687, JX462689, JX462691 *to* JX462693, JX462695 *to* JX462698, JX462700 *to* JX462703, JX462705 *to* JX462706, JX462708 *to* JX462739, JX488759, JX494787, JX508849 *to* JX508853, JX524225, ...

. . . , JX535003, JX669265, JX669269, JX669285, KC733248 *to* KC733255, KC733259 *to* KC733261, KC733263, KC733265 *to* KC733276, KC911619, KC993958, KC993967, KC993973, KF055303, KF161271, KF161307, KF161455, KF836133, KJ371984, KJ401945, KJ856831, KJ856840, KM007555, KP669005 *to* KP669006, KP688570 *to* KP688571, KP691018, KP691986, KP692737, KP698513 *to* KP698514, KP698575, KP702761, KP702821, KR864755, KR902533 *to* KR902539, KR919601, KT002469, KT006904, KT748522, KT756878, KT763050 *to* KT763051, KT764936, KT778765 *to* KT778766, KT779554, KU754494 *to* KU754496, KU757488, KU761989 *to* KU761990, KX079702 *to* KX079705, KX350100, KX437654 *to* KX437656, KX458252, KX459519, KX459697, KX460824, KX527571, KX530926, KX530929, KX530931 *to* KX530932, KX539224 *to* KX539226, KX557487