ANÁLISIS DE SECUENCIAS DE ADN MEDIANTE TÉCNICAS DE TRATAMIENTO DE DATOS MASIVOS

ANALYSIS OF DNA SEQUENCES USING BIG DATA TECHNIQUES

TRABAJO FIN DE GRADO
CURSO 2022-2023

AUTORES
GLORIA PORTO CABERO
SANDRA SÁNCHEZ PREVOST

DIRECTOR
JOSÉ IGNACIO REQUENO JARABO
JORGE ÁLVAREZ JARRETA

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

ANÁLISIS DE SECUENCIAS DE ADN MEDIANTE TÉCNICAS DE TRATAMIENTO DE DATOS MASIVOS

ANALYSIS OF DNA SEQUENCES USING BIG DATA TECHNIQUES

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTOR
GLORIA PORTO CABERO
SANDRA SÁNCHEZ PREVOST

DIRECTOR
JOSÉ IGNACIO REQUENO JARABO
JORGE ÁLVAREZ JARRETA

CONVOCATORIA: JUNIO 2023

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y ADMINISTRACIÓN DE EMPRESAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

AGRADECIMIENTOS

A José Ignacio Requeno Jarabo, Jorge Álvarez Jarreta y Fátima Díaz Da Corte por su guía y apoyo constante a lo largo de este proceso.

También, a Agustín Estrada Peña, catedrático de la Facultad de Veterinaria en la Universidad de Zaragoza, por su revisión del proyecto y aportaciones sobre líneas de trabajo futuro.

RESUMEN

Análisis de secuencias de ADN mediante técnicas de tratamiento de datos masivos

El presente Trabajo de Fin de Grado pretende explorar el uso de técnicas de procesamiento de datos masivos para el tratamiento y análisis de secuencias de ADN, con el propósito de proporcionar una visión global de las herramientas y entornos disponibles, la actualización de bibliotecas preexistentes y el diseño de un flujo de trabajo de referencia para la aplicación a un conjunto de datos concretos.

Palabras clave

ADN, ADN mitocondrial, Análisis secuencias, Datos masivos, Jupyter Notebook, Nextflow, Python, Alineamiento múltiple de secuencias

ABSTRACT

Analysis of DNA sequences using Big Data techniques

This project aims to explore the use of big data processing techniques for the treatment and analysis of DNA sequences, with the purpose of providing a comprehensive overview of the available tools and environments, the update of pre-existing libraries, and a reference workflow for the application to a specific database.

Keywords

DNA, Mitochondrial DNA, Sequence analysis, Big Data, Jupyter Notebook, Nextflow, Python, Multiple sequence alignment

ÍNDICE DE CONTENIDOS

TABLE OF CONTENTS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

El análisis de secuencias de ADN se ha convertido en una herramienta fundamental en la investigación en áreas como la genética, la biología molecular y la medicina. Sin embargo, la complejidad y la cantidad de datos obtenidos a través de las tecnologías de secuenciación masiva presentan un desafío para su procesamiento y análisis.

Este Trabajo de Fin de Grado explora la utilización de técnicas de tratamiento de datos masivos para el procesamiento y análisis de ADN, con el objetivo de ofrecer una visión general de las herramientas disponibles, la actualización de bibliotecas preexistentes y el diseño de un flujo de trabajo (o pipeline) para su aplicación a un conjunto de datos concreto. Más en detalle, durante este Trabajo de Fin de Grado se trabaja y actualiza MEvoLib, una biblioteca en Python con funcionalidades para la extracción y manipulación de secuencias de ADN. Después, se aplica la nueva versión de MEvoLib al diseño de un flujo de trabajo mediante el gestor de tareas Nextflow. Finalmente, se emplea el entorno desarrollado para el estudio del ADN de las garrapatas debido a su interés biológico, principalmente en su papel de vector en la transmisión de enfermedades patógenas.

Resulta importante comprender las implicaciones que supone trabajar con, en este caso, ADN mitocondrial de diferentes especies de garrapatas. La abundancia de ADN mitocondrial implica un mayor índice de variabilidad y mutación en la región hipervariante de dicho ADN, que puede resultar en complicaciones en el momento de la secuenciación. Asimismo, las secuencias mitocondriales aportan menos información para la identificación de especies e inferencia de relaciones que las secuencias nucleares, ya que la longitud de estas últimas es hasta cinco órdenes de magnitud mayor.

En el caso concreto de garrapatas, trabajar con ADN mitocondrial puede presentar ciertas limitaciones como la falta de información sobre la especie o la posibilidad de que haya procesos de selección concretos de relevancia en la evolución

de las secuencias. El presente capítulo reúne los términos, conceptos y objetivos esenciales del proyecto para comprender el proceso de recopilación y tratamiento masivo de datos para el análisis de secuencias de ADN y, en particular, para el contexto del ADN mitocondrial (un tipo especial de ADN).

Para aportar un contexto biológico previo al desarrollo de la disertación, cabe destacar que MEvoLib soporta la descarga y análisis de genes o secuencias de ADN independientemente de su fuente, sea este ADN de origen tanto mitocondrial como nuclear. Las definiciones de estos términos se pueden encontrar en el glosario de conceptos desarrollado en el Apéndice 1.

De esta manera, en la primera sección de este capítulo se desarrolla una introducción de las motivaciones de este trabajo, seguida por una segunda sección en la que se plasman los principales objetivos que se persiguen y, finalmente, una tercera sección en la que se plasma el plan de trabajo para su consecución.

1.1 Motivación y estado de la cuestión

La principal motivación de este proyecto es mejorar la utilización de técnicas de procesamiento de datos masivos para el análisis de secuencias de ADN mitocondrial. Para ello, dicha mejora implica la sistematización y actualización de herramientas de software preexistentes que optimicen la precisión y el rendimiento de los análisis de la filogenética.

Asimismo, con el enfoque del flujo de trabajo adaptado a las necesidades del usuario junto con la posibilidad de realizar múltiples consultas y tareas en paralelo, se persigue incrementar y mejorar el alcance de las diferentes técnicas de sistematización que encontramos en este campo de investigación.

De esta manera, se intenta proporcionar solución a las dificultades que surgen a gran escala, donde las herramientas de software comunes son ineficientes o, en el peor de los casos, no convergen. Con este fin, podemos distinguir tres puntos principales que motivan la realización de este trabajo:

- Actualizar el marco común que aporta la biblioteca MEvoLib para la mayoría de los estudios filogenéticos, de manera que el flujo de trabajo sea accesible tanto desde el campo de la informática como desde el campo de la biología.
- Mejorar el rendimiento e integración de la información disponible en las bases de datos para simplificar los pasos a seguir por parte del usuario.
- Unificar las diversas aplicaciones de la biblioteca MEvoLib en un flujo de trabajo que concentre todos los procedimientos y módulos con los que trabajar en los casos de uso específicos.

Para la consecución de dichos puntos, se ha realizado la correspondiente revisión de la literatura relacionada y, siguiendo la línea de trabajo futuro que se introduce en la tesis que presenta MEvoLib, "Análisis filogenético molecular: diseño e implementación de algoritmos escalables y fiables, y verificación de propiedades filogenéticas", por el Dr. Jorge Álvarez Jarreta (2017), se han marcado las pautas para, principalmente, mejorar el método de búsqueda, descarga y experiencia del usuario.

Además, durante los últimos seis años, la biblioteca no había experimentado ningún trabajo de mejora o mantenimiento, por lo que el estado de la cuestión previo al desarrollo del presente trabajo destaca por la falta de actualizaciones de la biblioteca en la mayoría de sus módulos así como en el soporte de las versiones mínimas de Python, 2.7 y 3.3, ambas ya obsoletas (la última se dejó de mantener hace más de 3 años).

1.2 Objetivos

Para cumplir con las motivaciones que plantean la ejecución de este trabajo, los objetivos previamente estipulados a cumplir se pueden clasificar de la siguiente manera:

- ☑ Configuración de la infraestructura de desarrollo.
- ☑ Actualización de la biblioteca MEvoLib.
- ☑ Desarrollo del flujo de trabajo en Jupyter Notebook usando la biblioteca MEvoLib.
- ☑ Desarrollo del flujo de trabajo en Nextflow usando la biblioteca MEvoLib.
- ☑ Validación del flujo de trabajo con un caso de estudio.
- ☑ Elaboración de la disertación del proyecto.

De este modo, la consecución de todos ellos bajo una metodología principalmente secuencial supone materializar las tres motivaciones esenciales y el objeto de trabajo explicado.

Además, para definir las acciones a tomar, se ha seguido un detallado plan de trabajo que garantizase el cumplimiento de cada pequeña tarea relacionada con alguno de los objetivos principales.

1.3 Plan de trabajo

Para la consecución de los objetivos descritos en la sección anterior, se plantea un plan de trabajo compuesto por una serie de tareas sucesivas siguiendo una metodología Agile, respaldada por la herramienta JIRA para facilitar el seguimiento de las distintas tareas, así como la sincronización entre las distintas partes y sus dependencias.

En una primera instancia, se realiza la configuración de los entornos necesarios en los que se va a desarrollar el presente trabajo. A su vez, se explica las razones y motivaciones para el uso de las distintas herramientas. Este apartado se verá en detalle en el capítulo 2 del presente documento.

En segundo lugar, se realiza una contextualización de la biblioteca MEvoLib y su estado más reciente en Python 2.7. Posteriormente, se procede a actualizar la biblioteca para adecuarla a una versión más reciente de Python. Para ello, se revisa el código existente en Python 2.7 y se adecua a los nuevos estándares de Python 3.10. Esta tarea se encuentra explicada en el capítulo 3.

En tercer lugar, tras la actualización a una versión más reciente de Python, se procede a implementar el flujo de trabajo con esta biblioteca para realizar el análisis de las secuencias de ADN. Este análisis se enfoca en el estudio de la composición genética de diferentes organismos, identificando las secuencias de nucleótidos presentes en el ADN e investigando las relaciones evolutivas entre organismos. Para poder llevarlo a cabo, es esencial integrar la fase de descarga de secuencias (Fetch), limpieza de datos y de información no pertinente, agrupación de los diferentes genes (Cluster) y alineamiento de las secuencias (Align) en un flujo automatizado para simplificar el procesamiento al usuario final. El flujo de trabajo permite la configuración de ciertos parámetros, como por ejemplo el nombre de los organismos o genes a estudiar. Cabe destacar que cada una de las etapas anteriormente mencionadas se corresponde con uno de los módulos proporcionados por la biblioteca MEvoLib, siendo la fase de alineamiento relevante para un estudio comparativo de las secuencias y la detección de posiciones del ADN con mutaciones.

Inicialmente, solo se implementa el flujo de trabajo de los módulos Fetch y Cluster utilizando la herramienta de Jupyter Notebooks. Luego, siguiendo con el Capítulo 4, se traslada el flujo de trabajo implementado a la herramienta Nextflow, donde, a su vez, se amplía la parametrización del mismo para permitir una mayor personalización del análisis de las secuencias y se añade el flujo de trabajo del módulo Align, actualizado por Fátima Díaz Da Corte.

En el capítulo 5, se recoge de forma breve los repositorios utilizados, así como el código fuente del presente proyecto.

Por otro lado, se destina el capítulo 6 para desarrollar del caso de uso principal del trabajo. Se explica de forma detallada el paso a paso para ejecutar un ejemplo del

flujo de trabajo de la biblioteca MEvoLib basado en la descarga, agrupación y alineamiento de genes mitocondriales de una especie de garrapata.

En el último capítulo del documento, se realiza una evaluación y análisis de los objetivos y se exponen las conclusiones. Además, se pone de manifiesto las limitaciones encontradas que pueden ser referencias para trabajos futuros.

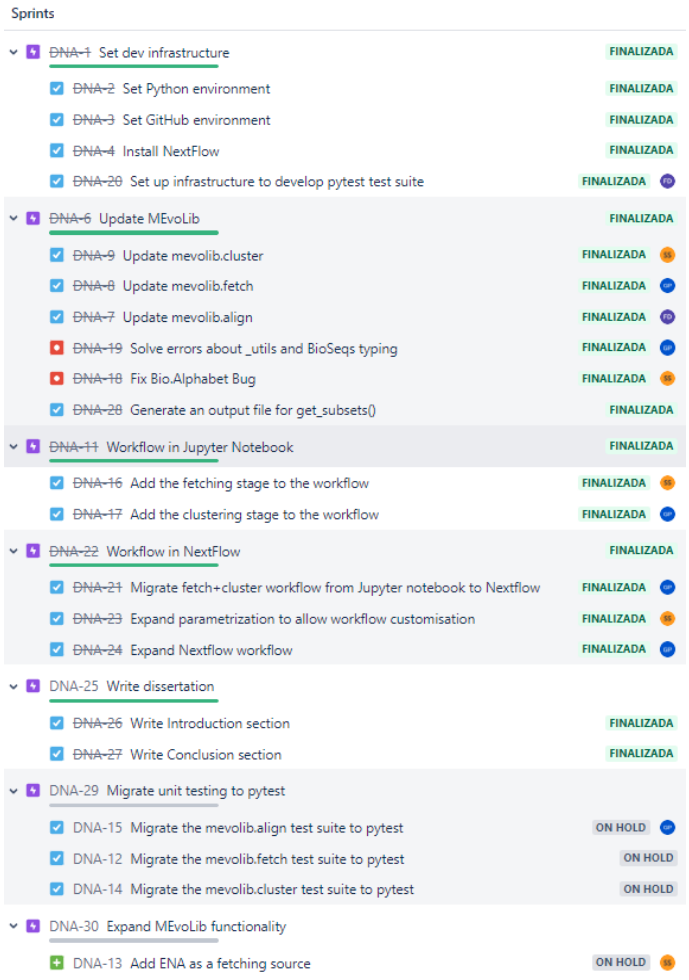
Por último, dentro del apartado de los Apéndices se recogen tanto el glosario de términos como un manual para la configuración e instalación de los entornos y herramientas utilizadas a lo largo del proyecto.

1.3.1 Gestión de tareas

Tal y como se cita anteriormente, para establecer un plan de trabajo ordenado se ha seguido una metodología Agile, respaldada por la herramienta JIRA para facilitar el seguimiento y sincronización de las distintas tareas.

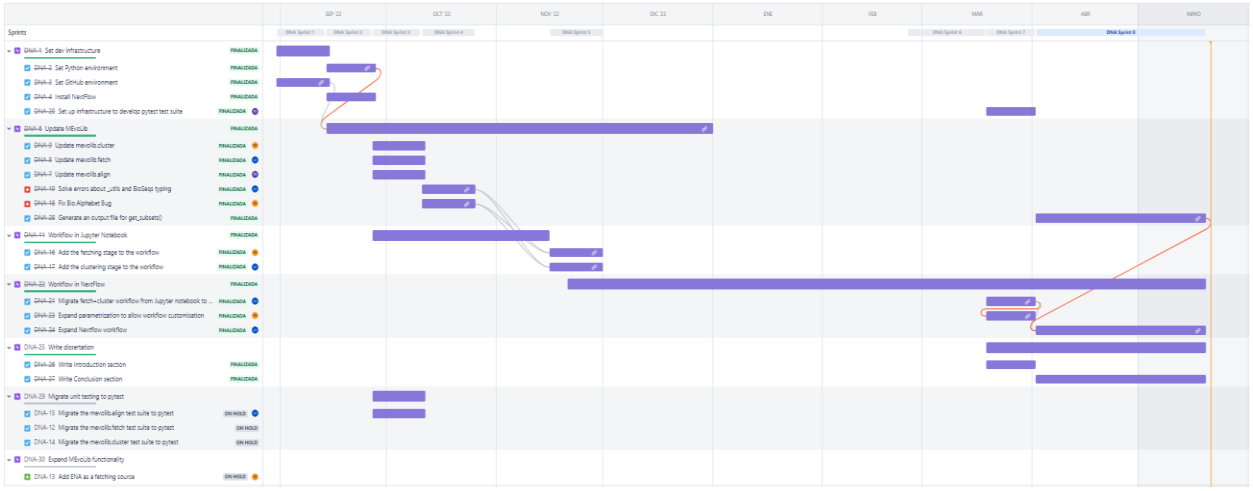
En las siguientes figuras se plasma la hoja de ruta seguida con cada tarea, las fechas correspondientes (desde septiembre de 2022 hasta mayo de 2023) e índices de esfuerzo reflejados como reuniones establecidas y líneas de código modificadas en el código fuente.

Figura 1: Hoja de ruta del plan de trabajo seguido



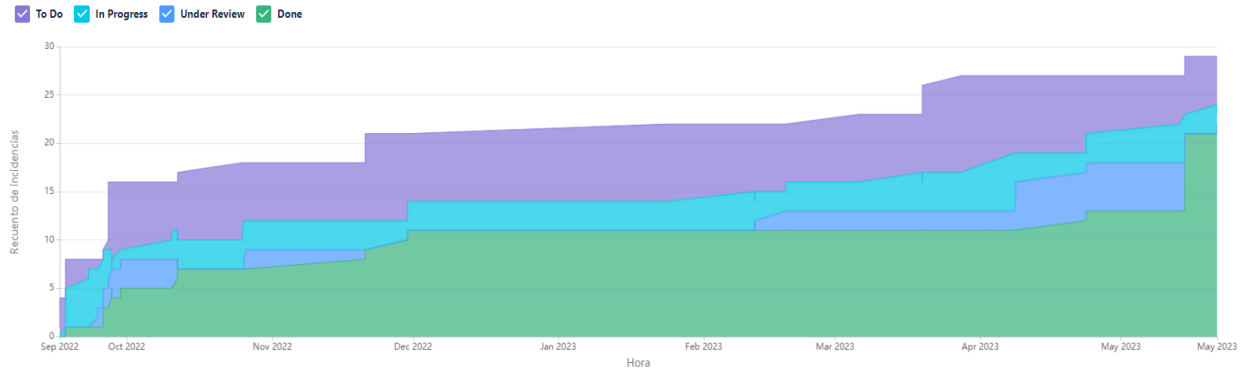
Fuente: Herramienta Jira (Atlassian, 2023).

Figura 2: Diagrama de Gantt de la hoja de ruta



Fuente: Herramienta Jira (Atlassian, 2023).

Figura 3: Diagrama de flujo acumulado



Fuente: Herramienta Jira (Atlassian, 2023).

Además, a lo largo de todo el proyecto, se han establecido reuniones cada dos semanas de una hora de duración, destinadas a mencionar el trabajo hecho hasta el momento, los posibles obstáculos encontrados y las siguientes acciones a implementar. Por tanto, se han realizando un total de 30 reuniones de equipo, lo que se traduce en 30 horas destinadas a este fin.

Por otro lado, contrastando el código final de la biblioteca con el repositorio original de MEvoLib desde GitHub, se han realizado un total de 122 commits, abarcando 110 archivos de la biblioteca y, en ellos, modificando un total de 3731 líneas de código.

Por útlimo, el repositorio correspondiente al código fuente del flujo de trabajo, parasitology, se ha desarrollado desde cero, con un total de 435 líneas de código.

Chapter 1' - Introduction

DNA sequence analysis has become a fundamental tool in research areas such as genetics, molecular biology and medicine. However, the complexity and amount of data obtained through mass sequencing technologies present a challenge for its processing and analysis.

This Final Degree Project explores the use of massive data processing techniques for DNA processing and analysis, with the aim of offering an overview of the available tools, updating pre-existing libraries and design a workflow for their application to a specific dataset.

During this project MEvoLib, a Python library with functionalities for the extraction and manipulation of DNA sequences, is worked on and updated. Then, we apply the new version of MEvoLib to a workflow design using the Nextflow task manager. Finally, we used the developed environment to study tick DNA due to its biological interest, particularly its role as vectors in the transmission of pathogenic diseases.

On the other hand, it is important to understand the implications of working with, in this case, mitochondrial DNA from different tick species. The abundance of mitochondrial DNA implies a higher rate of variability and mutation in the hypervariable region of that DNA, which can lead to complications during sequencing. Additionally, mitochondrial sequences provide less information for species identification and inference of relationships than nuclear sequences, as the length of the latter is up to five orders of magnitude greater.

In the specific case of ticks, working with mitochondrial DNA may present certain limitations, such as the lack of information about the species or the possibility that there are specific selection processes relevant to the evolution of the sequences. This chapter brings together the essential terms, concepts, and objectives of the project to understand the process of massive data collection and processing for DNA sequence analysis, particularly in the context of mitochondrial DNA (a special type of DNA).

To provide a biological context before the development of the dissertation, it should be noted that MEvoLib supports the download and analysis of genes or DNA sequences regardless of their source, whether it is mitochondrial or nuclear DNA. The definitions of these terms can be found in the glossary of concepts developed in Appendix 1.

Thus, the first section of this chapter provides an introduction to the motivations of this work, followed by a second section outlining the main objectives to be pursued, and finally, a third section presents the work plan for achieving these objectives.

1.1' Motivation and state of the art

The main motivation of this project is to enhance the use of Big Data processing techniques for the analysis of mitochondrial DNA sequences. To achieve this, the improvement implies the systematization and updating of pre-existing software tools that optimize the precision and performance of phylogenetic analyses.

Furthermore, with the approach of the workflow adapted to the user's needs together with the possibility of carrying out multiple queries and tasks in parallel, the aim is to increase and improve the scope of the different systematization techniques found in this research field.

In this way, it is intended to provide a solution to the difficulties that arise on a large scale, where common software tools are inefficient or, in the worst case, do not converge. To this end, we can distinguish three main points that motivate the realization of this work:

• Update the common framework provided by the MEvoLib library for most phylogenetic studies, so that the workflow is accessible from both the field of informatics and the field of biology.

• Improve the performance and integration of the information available in the databases to simplify the steps to be followed by the user.

• Unify the various applications of MEvoLib library into a workflow that concentrates all the procedures and modules with which to work in specific use cases.

In order to achieve these points, the corresponding literature review has been carried out and, considering future work that is introduced in the thesis which presents MEvoLib, "Molecular phylogenetic analysis: design and implementation of scalable and reliable algorithms and verification of phylogenetic properties", by Dr. Jorge Álvarez Jarreta (2017), guidelines have been set to, mainly, improve the search method, download and user experience.

In addition, during the last six years, the library had not undergone any improvement or maintenance work so, the state of the art prior to the development of this work stands out for the lack of updates to the library in most of its modules and, firstly, in the Python language version implemented, specifically supporting up to version 2.7 and 3.3.+.

1.2' Goals

To comply with the motivations that arise from the execution of this work, the previously stablished objectives to be met can be classified as follows:

- ☑ Configuration of the development infrastructure.
- ☑ MEvoLib library update.
- ☑ Development of the workflow in Jupyter Notebook using MEvoLib library.
- ☑ Development of the workflow in Nextflow using MEvoLib library.
- ☑ Validation of the workflow with a case study.
- ☑ Elaboration of the project dissertation.

Thus, the achievement of all of them under a mainly sequential methodology, materializes the three essential motivations and the object of explained work.

Additionally, to define the actions to be taken, a detailed work plan has been followed to guarantee compliance with each small task related to one of the main objectives.

1.3' Work plan

To achieve the objectives described in the previous section, a work plan is proposed, consisting of a series of successive tasks following an Agile methodology, supported by the JIRA tool to facilitate the tracking of different tasks, as well as the synchronization between different parts and their dependencies.

Firstly, the configuration of the necessary environments in which the present work will be developed is carried out. The reasons and motivations for using different tools are also explained. This section will be detailed in Chapter 2 of this document.

Secondly, a contextualization of the MEvoLib library and its most recent state in Python 2.7 is presented. Subsequently, the library is updated to adapt it to a newer version of Python. To do this, the existing code in Python 2.7 is reviewed and adjusted to meet the new standards of Python 3.10. This task is explained in Chapter 3.

Thirdly, after updating to a more recent version of Python, the workflow of the library is implemented to perform the analysis of DNA sequences. This analysis focuses on studying the genetic composition of different organisms, identifying nucleotide sequences present in DNA, and investigating evolutionary relationships between organisms. To carry it out, it is essential to integrate the sequence downloading phase (Fetch), data cleaning, removal of irrelevant information, gene clustering (Cluster), and sequence alignment (Align) into an automated workflow to simplify processing for the end user. The workflow allows for the configuration of certain parameters, such as the names of organisms or genes to study. It should be noted that each of the aforementioned stages corresponds to one of the modules provided by the MEvoLib library, with the alignment phase being relevant for comparative sequence analysis and detection of DNA positions with mutations.

Initially, only the workflow of the Fetch and Cluster modules is implemented using Jupyter Notebooks. Then, following Chapter 4, the implemented workflow is transferred to the Nextflow tool, where the parameterization of the workflow is expanded to allow for greater customization of sequence analysis, and the workflow of the Align module, updated by Fátima Díaz Da Corte, is added.

Chapter 5 briefly discusses the repositories used, as well as the source code of the present project.

On the other hand, Chapter 6 is devoted to developing the main use case of the work. A detailed step-by-step explanation is provided for executing an example of the MEvoLib library's workflow based on the downloading, clustering, and alignment of mitochondrial genes from a tick species.

In the last chapter of the document, an evaluation and analysis of the objectives are carried out, and the conclusions are presented. In addition, the encountered limitations that can serve as references for future work are highlighted.

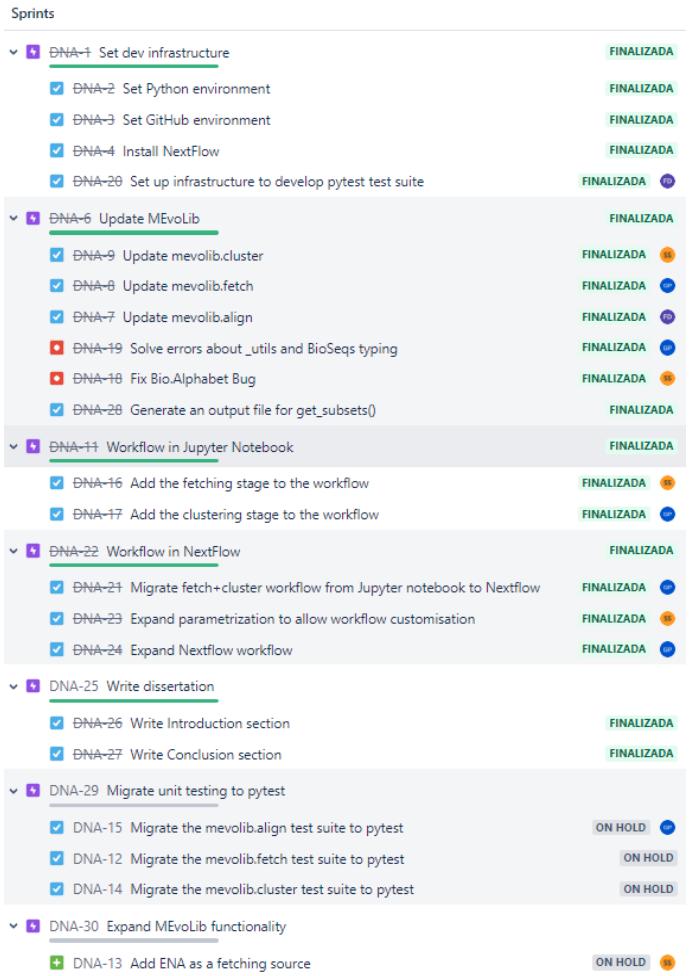
Finally, the Appendices section includes both the glossary of terms and a manual for the configuration and installation of the environments and tools used throughout the project.

1.3.1' Task management

As mentioned earlier, to establish an organized work plan, an Agile methodology has been followed, supported by the JIRA tool to facilitate tracking and synchronization of different tasks.

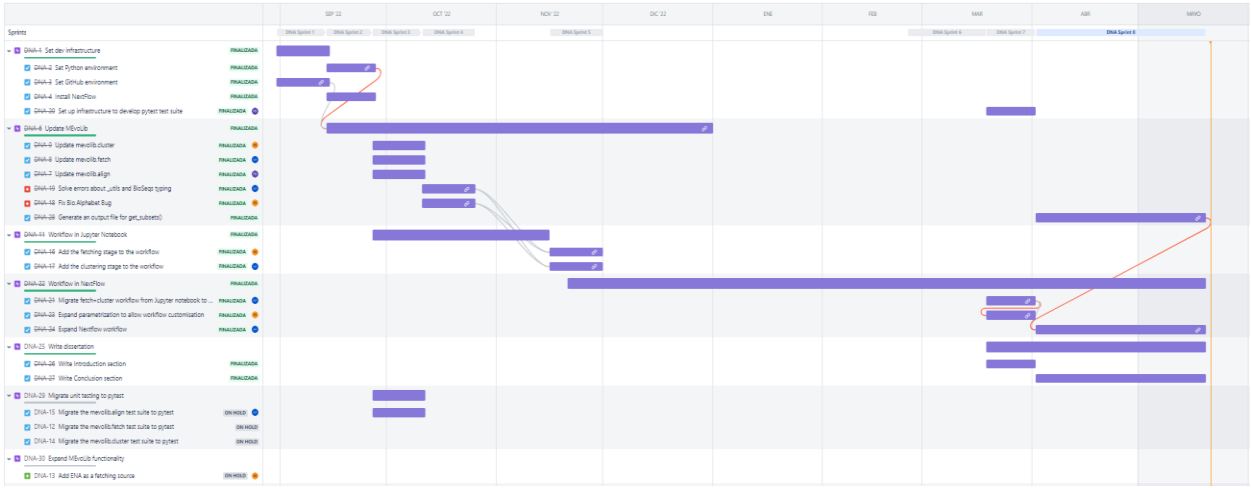
The following figures depict the roadmap followed with each task, the corresponding dates (from September 2022 to May 2023), and effort indicators reflected as established meetings and lines of code modified in the source code.

Figura 4: Roadmap of the work plan followed



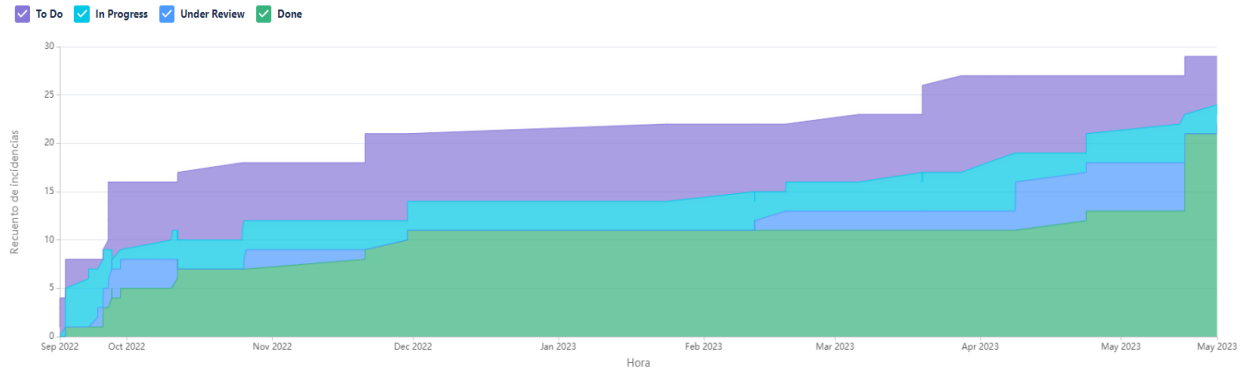
Source: Jira tool (Atlassian, 2023).

Figura 5: Gantt chart of the roadmap



Source: Jira tool (Atlassian, 2023).

Figura 6: Cumulative flow diagram



Source: Jira tool (Atlassian, 2023).

In addition, throughout the project, meetings have been scheduled every two weeks, with a duration of one hour each, to discuss the work done so far, potential obstacles encountered, and the next steps to be implemented. Therefore, a total of 30 team meetings have been conducted, amounting to 30 hours dedicated to this purpose.

Furthermore, comparing the final code of the library with the original MEvoLib repository on GitHub, a total of 122 commits have been made, covering 110 files in the library and modifying a total of 3731 lines of code.

Lastly, the repository corresponding to the source code of the workflow, "parasitology," has been developed from scratch, consisting of a total of 435 lines of code.

Capítulo 2 - Infraestructura SetDev

Antes de comenzar con el desarrollo de la disertación, conviene mencionar la motivación de uso de las herramientas que se han configurado para cumplir con el objeto de estudio del presente trabajo.

En primer lugar, la elección del lenguaje Python viene previamente dada por la ya existente biblioteca MEvoLib, la cual, en sus inicios, contempla este lenguaje de alto nivel debido a su facilidad de uso, flexibilidad y gran cantidad de bibliotecas disponibles, en este caso concreto relacionadas con la bioinformática.

En segundo lugar, el uso de Pyenv se justifica con la gestión de múltiples versiones de Python en un mismo sistema, dada la motivación de actualizar la biblioteca a la versión Python 3.10.6. De este modo, se puede cambiar fácilmente entre versiones de Python y administrar los paquetes y dependencias específicas para cada proyecto sin mayor dificultad.

Del mismo modo, Pyenv-Virtualenv permite crear los entornos virtuales aislados para proyectos específicos, con su propia versión de Python y sus propias bibliotecas. Esto resulta de gran utilidad en este proyecto para evitar conflictos de dependencias y distinguir los entornos necesarios.

Además, en vista a realizar contribuciones en el repositorio original de la biblioteca, se utiliza GitHub como sistema de control de versiones, rastreando los cambios en el código fuente del proyecto a lo largo del tiempo. Cabe destacar que las contribuciones a la rama correspondiente se pueden hacer indistintamente entre desarrolladores colaborando y compartiendo las actualizaciones pertinentes.

En este mismo hilo, se emplea el editor de código fuente Visual Studio Code, el cual, gracias a la multitud de extensiones y personalización que presenta, resulta una herramienta de alta calidad. Además, su sencilla y completa integración con GitHub hace de este editor una aplicación clave para la gestión y colaboración es las distintas versiones de código fuente y módulos.

En cuanto al desarrollo del flujo de trabajo, en primer lugar se opta por desarrollar una previa versión inicial en Jupyter Notebook, debido a su sencillez e interfaz gráfica frente a Nextflow. Como se indica, Jupyter Notebook proporciona una plataforma interactiva en la que integrar código, ejecución, resultados, comentarios y visualización del proceso de descarga y análisis de las secuencias en un único documento haciendo que este sea fácil de compartir y reproducir.

Seguidamente, Nextflow es una herramienta con matices más complejos que Jupyter Notebook pero cuyo alcance es mayor en términos de gestión y automatización del flujo de trabajo de Fetch, Cluster y Align, ya que permite desplegarlo y reproducirlo en varios entornos como servidores en la nube o computación local. De este modo, gestionando las tareas, recursos y dependencias de forma modular resulta más útil y eficiente para el desarrollo y ejecución del flujo de trabajo completo de MEvoLib.

A su vez, se emplea la base de datos GenBank para la extracción de secuencias, ya que es una base de datos de acceso público que permite almacenar y distribuir datos de secuenciación. Es gestionada por el Centro Nacional para la Información Biotecnológica (NCBI). Su uso viene motivado por el aprovechamiento de los recursos existentes, ya que esta herramienta se encuentra implementada en la versión original de la biblioteca MEvoLib. En esta línea, se considera la incorporación de la base de datos ENA (European Nucleotide Archive) para ampliar los recursos disponibles de MEvoLib. Sin embargo, por el alcance del presente proyecto, no se termina implementando.

Por último, la motivación principal de considerar Mafft en el módulo de Align como técnica de alineamiento múltiple de secuencias reside en la elección de una herramienta rápida y precisa para la alineación de las secuencias biológicas obtenidas tras la ejecución de los módulos anteriores, de la manera más eficiente posible. Para lograrlo, Mafft utiliza una variedad de técnicas avanzadas, como la optimización iterativa, la alineación progresiva y la alineación de guía, para obtener alineamientos de alta calidad. Además, es una de las herramientas más usada en la comunidad bioinformática para realizar alineamientos de secuencias biológicas.

Capítulo 3 - MEvoLib

Tal y como se menciona en la introducción de este proyecto, y siguiendo las afirmaciones de Álvarez-Jarreta y Ruiz-Pesini (2016), los estudios centrados en la evolución molecular involucran problemáticas computacionales cuya resolución, en la mayoría de los casos, se aproxima de manera heurística.

Este capítulo presenta la versión original de MEvoLib y las nuevas mejoras introducidas. De este modo, MEvoLib es la primera biblioteca de evolución molecular desarrollada para el lenguaje Python, proporcionando, así, un marco de trabajo consolidado con diversas herramientas y métodos comunes de los flujos de trabajo relacionados con la evolución molecular.

A diferencia de las bibliotecas de bioinformática ya existentes, MEvoLib se centra en las etapas involucradas en los estudios de evolución molecular, consolidando el conjunto de herramientas de propósito común en una única interfaz de alto nivel con acceso rápido y parametrizaciones frecuentes.

De este modo, el agrupamiento de genes a partir de secuencias parciales o, en el mejor de los casos, completas, se ha mejorado con la implementación de un método que aporta información externa accesible. Así, la información y conocimientos de los investigadores volcados en los datos de características de GenBank (base de datos de secuencias genéticas del NCBI) pueden ser procesados y empleados para optimizar el proceso de secuenciación.

Además, con el objetivo de aprovechar correctamente los recursos del usuario, se han considerado ajustes y optimizaciones del proceso de conexión y obtención de información a partir de las bases de datos del Centro Nacional para la Información Biotecnológica de Estados Unidos, NCBI, junto con las paralelizaciones implementadas en vista a escalar la herramienta a proyectos de mayor tamaño.

3.1 MEvoLib inicial

El funcionamiento de la biblioteca MEvoLib se apoya en diversas clases y módulos implementados en BioPython (Cock *et al.*, 2009). De esta manera, MEvoLib cubre las tareas principales involucradas en los estudios de evolución molecular, incluyendo así: la obtención y agrupación de datos apoyados con el conocimiento biológico disponible, la alineación múltiple de secuencias, la inferencia filogenética y el ensamblaje filogenético.

En la siguiente tabla se representan los módulos de MEvoLib según la categoría de trabajo a la que pertenece.

Tabla 1: Procesos y módulos de MEvoLib

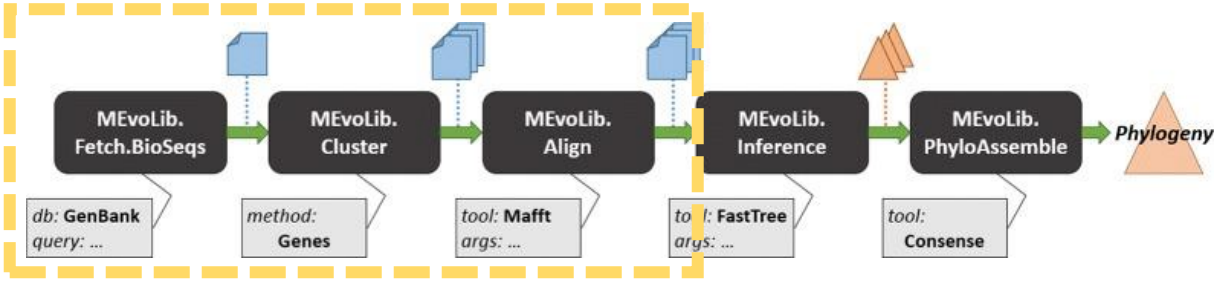
Proceso	Módulos
Data	rCRS
Fetch	BioSeqs, PhyTrees
Cluster	Naïve rows, Naïve colums,
	PRD, Genes
Align	Mafft, Clustal Omega, Muscle
Inference	FastTree, RAxML
PhyloAssemble	Consense

Fuente: Álvarez-Jarreta y Ruiz-Pesini, 2016

Asimismo, el flujo de trabajo con la herramienta de MEvoLib desde la recopilación (Fetch) de los datos, la agrupación de los mismos (Cluster) y su respectivo alineamiento (Align) con la herramienta especificada (en este caso, Mafft) se puede plasmar en un ejemplo de uso motivador como encontramos en la Figura 1. Además, se puede observar que el flujo de trabajo mencionado puede ampliarse a la inferencia

filogenética y a la construcción del árbol correspondiente tras el alineamiento de las secuencias.

Figura 7: Ejemplo de flujo de trabajo con la biblioteca MEvoLib



Fuente: Álvarez-Jarreta y Ruiz-Pesini, 2016

De este modo, cada etapa presenta la interfaz de MEvoLib para cada procedimiento mencionado, junto con una posible parametrización. Los conjuntos y subconjuntos de secuencias se han representado por hojas, y las filogenias resultantes se pueden ver representadas por triángulos.

MEvoLib.Fetch

Primeramente, **MEvoLib.Fetch** es el módulo utilizado para buscar y recopilar información biológica de archivos locales y de diferentes bases de datos del NCBI, por ejemplo GenBank (Benson et al., 2010).

La principal ventaja de uso de este módulo es la fusión de datos de más de una fuente, generando un único archivo uniforme con la información relevante obtenida durante el proceso de recopilación. Además, la herramienta añade un controlador de recuperación por lotes y una función de actualización para el módulo de Entrez proporcionada por BioPython y mejorando, así, el proceso de interacción con la Base de Datos.

Este módulo se divide en dos clases diferenciadas:

- ☑ BioSeqs, para buscar secuencias biológicas a partir de cualquiera de las dos fuentes mencionadas.
- ☑ PhyTrees, para buscar árboles filogenéticos únicamente entre archivos locales.

Ambas clases pueden trabajar con cualquier formato soportado por BioPython, y una vez recopilada la información deseada, los datos se vuelcan en archivos de formato GENBANK o NEWICK, respectivamente. Además, cada archivo generado cuenta con un informe de reporte en el que se especifican el número de secuencias o árboles guardados y una lista detallada de la fuente, fecha, hora e instrucciones de búsqueda usadas.

De este modo, BioSeqs implementa tres propiedades características para la obtención de información de las bases de datos del NCBI:

- Limitación del tamaño del conjunto de secuencia obtenidas de la base de datos, según la consulta dada, de manera que el usuario pueda obtener únicamente una muestra representativa, en caso de que así lo desee.
- Cálculo de un límite superior para el tamaño esperado de la secuencia, en bytes, a partir de las instrucciones del propio NCBI. Además, con la limitación del tiempo máximo de descarga para cada lote, se consigue optimizar el intérvalo y el número de solicitudes realizadas, de forma que el proceso de descarga de grandes bases de datos resulte más eficiente y evite pérdidas de datos ante errores de comunicación inesperados.

- Eliminación de secuencias no presentes en el conjunto de la base de datos del NCBI, recuperando así únicamente aquellas secuencias que son nuevas o revisadas, mejorando, de esta manera, el proceso de actualización.

Asimismo, con la incorporación de información biológica en los métodos de MEvoLib para afinar la precisión de la herramienta, se consigue evitar conexiones a Internet innecesarias para recuperar los datos en tiempo de ejecución, evitando errores y acelerando los métodos de la biblioteca.

MEvoLib.Cluster

En segundo lugar, el módulo **MEvoLib.Cluster** se centra en los procedimientos de preprocesamiento relacionados con la división del conjunto de datos de entrada en subconjuntos más pequeños.

Esta interfaz se ha diseñado para proporcionar cuatro enfoques de división diferenciados: naïve rows, naïve columns, PRD y genes. Varios de estos enfoques contemplan técnicas de paralelización para mejorar su rendimiento, especialmente en escenarios a gran escala. Además, cada método devuelve el agrupamiento como un diccionario, con identificadores establecidos como claves y una lista de objetos SeqRecord (BioPython) como valores para cada una de ellas.

Según el criterio de agrupación de cada método, podemos diferenciarlos de la siguiente manera:

i. Criterio basado en aplicar la estrategia de Divide y Vencerás para cumplir las técnicas de paralelización en los siguientes procesos. La base de datos se procesa como una matriz de caracteres **n**x**m** (en caso de discrepancias de tamaño, se añaden espacios al final) y se implementa con los métodos naïve:

    a. Naïve rows: divide la matriz en un número dado de conjuntos (**k**) no contiguos. Para aprovechar una salida balanceada, el método genera **j** sets de **n**/**k** secuencias y **k**-**j** sets de **n**/**k** secuencias también, siendo **j** = (**n** mod **k**)

    b. Naïve columns: calcula la longitud del fragmento y los índices de rango basándose en la secuencia de entrada más larga, **m**, y el número dado de conjuntos, **k**. Este algoritmo aplica las mismas ecuaciones que el algoritmo anterior, pero, en este caso, reemplazando **n** por **m**.

ii. Criterio basado en el aprovechamiento del conocimiento biológico para mejorar la precisión de los siguientes estados, dividiendo los datos en

subconjuntos cuya relación, en término biológicos, es mayor. Este criterio se implementa con los métodos:

a. Descomposición PRD: descomposición DCM3 recursiva acolchada DACTAL (Nelesen *et al.*, 2012). Con este tipo de descomposición, se extrae un conjunto de subconjuntos superpuestos de filogenia, donde la unión de dichos subconjuntos es igual al conjunto de datos de entrada. La entrada de este método la conforman el conjunto de datos, la filogenia, el tamaño máximo de cada subconjunto y el número de secuencias superpuestas. Cada subconjunto cuyo tamaño no cumple la condición especificada se usa nuevamente como entrada (con su correspondiente subárbol) para el proceso de descomposición del PRD. Además, en MEvoLib se mejora este método reemplazando su parte recursiva por un procedimiento iterativo paralelizado donde los conjuntos que requieren una mayor descomposición son gestionados mediante una cola.

b. Descomposición Genes: este método ha sido diseñado para generar tantos conjuntos como genes disponibles haya en las secuencias de entrada. La idea principal de este enfoque es aprovechar el conocimiento biológico para extraer todos los genes de un conjunto de secuencias de ADN. De esta manera, MEvoLib incorpora el método Genes para extraer todos los genes de un conjunto de secuencias, ya que las bases de datos del NCBI almacenan este conocimiento biológico como metadatos en cada secuencia (con una representación similar a XML). Este método, además, admite filtrado por identificadores de etiquetas para extraer así las características concretas en las que el usuario está interesado. Cabe destacar que el método también cuenta con una ecuación de muestreo estadístico que permite detectar aquellos términos relacionados para el mismo gen con una baja representación de muestreo, indicando, en el archivo de registro, aquellos términos inciertos y la lista de secuencias

que los respaldan. De este modo, el usuario también puede detectar las secuencias que fusionan dos genes debido a un error en su información.

MEvoLib.Align

En tercer lugar, el módulo de alineación de secuencias múltiples (o multiple sequence alignment, MSA, en inglés) se enfoca en trabajar con diversas herramientas para obtener el alineamiento buscado.

La característica principal de esta interfaz es que presenta una única parametrización aplicable para cada herramienta MSA incluida en MEvoLib. De esta manera, el usuario ha de indicar la herramienta que desea ejecutar, el archivo de secuencias de entrada, el formato del archivo y la lista de argumentos de la herramienta.

La alineación resultante, para todas las herramientas, se devuelve como un objeto MultipleSeqAlignment (BioPython) y, a mayores, se puede agregar un archivo de salida a la lista anterior de parámetros para guardar el MSA resultante.

Este propósito de archivos de entrada-salida es doble, ya que se busca garantizar la lectura y escritura de dichos formatos y, también, se persigue aplicar la conversión de formato si estos no son compatibles con la herramienta MSA especificada. Ambos objetivos facilitan el trabajo del usuario y mejora la interfaz en escenarios donde el siguiente proceso requiere un formato de archivo específico.

Para cada herramienta MSA contemplada en la biblioteca de MEvoLib se ha creado un diccionario de configuración donde las parametrizaciones comunes se referencian mediante una palabra clave que se puede pasar a la interfaz de Align directamente y por defecto, en lugar de introducir la lista de argumentos. El usuario

puede añadir más palabras clave y configuraciones en el archivo de herramientas correspondiente.

En la primera versión de MEvoLib se incluyen como herramientas MSA: Mafft (Katoh *et al.*, 2002), Clustal Omega (Sievers *et al.*, 2011) y MUSCLE (Edgar, 2004). Además, para los usuarios más experimentados, este módulo puede iniciar herramientas MSA no incluidas en la versión instalada de la biblioteca.

MEvoLib.Inference

Esta interfaz maneja el proceso de inferencia filogenética con herramientas de software comúnmente utilizadas. Al igual que el módulo de Align, Inference presenta una parametrización única para cada herramienta incluida en la biblioteca de MEvoLib.

Para su funcionamiento, el usuario debe proporcionar la herramienta de estimación filogenética, el archivo de secuencia de entrada, el formato de entrada y la lista de argumentos de la herramienta.

La filogenia resultante se devolverá en un objeto Tree (BioPython) junto con su puntuación de verosimilitud logarítmica. La lista de parámetros se puede ampliar para incluir el archivo de árbol de salida y su formato, donde se guardará el árbol filogenético.

Además, Inference, además del diccionario de configuraciones comunes para cada herramienta, también incluye la funcionalidad de conversión automática de formato, lo cual facilita la interacción de MEvoLib con otros métodos y herramientas de software.

Las herramientas incluidas de inferencia filogenética en la primera versión de MEvoLib son: FastTree (Price *et al.*, 2009) y RAXML (Stamatakis, 2006).

MEvoLib.PhyloAssemble

Por último, este módulo de MEvoLib se centra en la implementación del proceso de estimación de árboles de consenso. Para la obtención del árbol de consenso de un conjunto de árboles filogenéticos, la interfaz requiere especificar: la herramienta a emplear, el archivo de árbol de entrada, el formato de archivo y la lista de argumentos de la herramienta.

El árbol de consenso resultante se devuelve en un objeto Tree (BioPython). Además, PhyloAssemble también incluye la funcionalidad de conversión automática de formato.

La primera versión de MEvoLib incorpora una versión modificada del método Consense de PHYLIP (Felsenstein, 2006) para el proceso de estimación del árbol de consenso. La implementación de esta versión modificada de Consense permite incorporar dos nuevos argumentos: las rutas de los archivos de entrada y las rutas de los archivos de salida, para así evitar posibles problemas de ejecución en infraestructuras como clústeres.

3.2 Actualización de MEvoLib

La biblioteca de MEvoLib no ha sido actualizada desde su primera versión, lanzada en 2016. De este modo, en sus inicios fue desarrollada para soportar Python 2.7 y 3.3+, pero las nuevas actualizaciones y funcionalidades añadidas en Python 3.10+ actualmente no se contemplan en la biblioteca, lo cual es un área de mejora que se pretende explotar en este proyecto.

Las actualizaciones se han centrado en el copyright y los módulos de Fetch, Cluster y Align. En los párrafos siguientes, se detalla en profundidad las modificaciones introducidas,

Primeramente, los archivos de licencia de copyright se han actualizado junto con los términos y condiciones de uso, reproducción y distribución, utilizando como referencia la página oficial de Apache (versión 2).

En primera instancia, el primer módulo del que analizaremos la actualización es **MEvoLib.Fetch**, para seguir el orden del flujo de trabajo.

Para esta primera revisión y actualización del código, el archivo __init__.py correspondiente a Fetch se ejecuta automáticamente, de modo que todas las importaciones del módulo correspondiente y de los módulos importados en él deben ser tratadas como importaciones absolutas (desde la raíz del paquete), en lugar de relativas (desde el presente directorio o de sus padres).

Del mismo, modo, la clase BioSeqs se adapta a la nueva licencia actualizada, y cada línea de su código es revisada y adaptada a la versión de Python 3.10, como se puede ver en las referencias adjuntadas en el apartado referente al código fuente.

Asimismo, las funciones principales que se han visto modificadas en mayor medida del módulo de MEvoLib.Fetch.BioSeqs han sido:

Tabla 2: Principales funciones actualizadas del módulo MEvoLib.Fetch.BioSeqs

Función	Descripción
`_get_entrez_db_rettype`	Comprobación para verificar que la base de datos dada está soportada en la clase BioSeqs, comparándola con el diccionario correspondiente
`_estimate_batch_size`	Estimación del tamaño del lote de procesamiento, considerando una velocidad de descarga de 1 Mbps y un tiempo máximo de búsqueda deseable de 5 minutos por lote

`__init__`	Función de inicialización de la clase encargada de crear el objeto BioSeqs con la información del diccionario de secuencias y su archivo de reporte asociado
`from_bioseqs` y `from_seqfile`	Creación del objeto BioSeqs recuperando toda la información de los archivos de secuencia y de informe previamente guardados
`from_entrez`	Creación del objeto BioSeqs a partir de la búsqueda de las secuencias coincidentes con la consulta correspondiente a la base de datos proporcionada del NCBI
`__len__`	Cálculo del número total de secuencias
`__str__`	Impresión de las secuencias de manera formalizada
`include`	Añade la información del archivo de secuencia al objeto BioSeqs
`join`	Función empleada para la unión de dos objetos BioSeqs
`update`	Actualización del objeto BioSeqs con los últimos valores incluidos en la base de datos del NCBI
`write`	Guardado de todas las secuencias relativas al objeto BioSeqs en un archivo con formato GENBANK (.gb), además del correspondiente archivo de reporte
`statistics`	Cálculo del número total de secuencias guardadas junto con su media, desviación, mínimos y máximos

Fuente: Elaboración propia

Como se ha mencionado previamente, para una visión más detallada de los cambios realizados, se pueden analizar los commits realizados en los repositorios de GitHub enlazados en el apartado de código fuente.

En segundo lugar, el siguiente módulo actualizado es **MEvoLib.Cluster**. Tras dicha revisión, el archivo __init__.py correspondiente a este módulo se ejecuta automáticamente para descomprimir la secuencia de ADN y el nombre del archivo FASTA de destino.

Dentro de este módulo, la clase Genes se adapta a la nueva licencia actualizada, y cada línea de su código es revisada y adaptada a la versión de Python 3.10+, (ver referencias del apartado de código fuente).

En resumen, las principales funciones implicadas en la actualización de la clase de este módulo han sido:

Tabla 3: Principales funciones actualizadas del módulo MEvoLib.Cluster.Genes

Función	Descripción
`_normalization`	Normaliza el input de la secuencia con la referencia de la misma
`_string_filter`	Devuelve el input introducido en la base de datos con las correcciones pertinentes para adaptarlo al formato de GenBank
`get_features`	Devuelve una lista con todas las palabras clave que se pueden encontrar en cualquier registro de consulta de GenBank
`map_seqs`	Empalma los genes de la lista de secuencias dada

Fuente: Elaboración propia

Por otro lado, la principal función actualizada de las clases **MEvoLib.Cluster.NaiveCols** y **MEvoLib.Cluster.NaiveRows** es `map_seqs`, en la que se concatenan las secuencias de ADN de la lista dada según el número de conjunto dados según las columnas o las filas, respectivamente.

Además, en cuanto a la clase **MEvoLib.Cluster.PRD**, las únicas dos funciones presentes y cuyo código ha sido revisado y actualizado son `_prd_decomposition` y `map_seqs`, con las que, primeramente, se aplica la descomposición PRD propia del sistema DACTAL al archivo del árbol con la superposición dada, tal y como se especificaba anteriormente y, a continuación, se genera un mapa de las secuencias en conjuntos.

En tercer lugar, se actualiza el módulo de **MEvoLib.Align**, por Fátima Díaz Da Corte, en vista a continuar adaptando el conjunto de la biblioteca.

Tabla 4: Principales funciones actualizadas del módulo MEvoLib.Align

Función	Descripción
`get_tools`	Devuelve la lista de herramientas de alineamiento incluidas en la versión actual de MEvoLib
`get_keywords`	Devuelve un diccionario con las palabras clave y sus argumentos correspondientes
`get_alignment`	Devuelve las secuencias alineadas

Fuente: Elaboración propia

Cabe destacar que la actualización de este módulo se ha enfocado en un único software de alineamiento múltiple de secuencias, Mafft. Este software utiliza técnicas basadas en la transformada de Fourier, herramienta matemática utilizada para analizar señales en el dominio de una frecuencia. En este contexto, el empleo de esta metodología permite comparar las secuencias de manera más rápida y eficiente.

A pesar de que en este trabajo se ha focalizado en el empleo de Mafft, la biblioteca también cuenta con otras dos herramientas MSA, concretamente Clustal Omega y Muscle, con las cuales se puede trabajar especificándolo en los argumentos y siguiendo la misma parametrización con la que se utiliza Mafft.

Capítulo 4 - Workflow

En el presente proyecto, el desarrollo de un flujo de trabajo efectivo es esencial para garantizar la correcta ejecución del plan del proyecto y alcanzar los resultados esperados.

Las dos principales herramientas para ejecutar este tipo de flujos de trabajos de datos que se han contemplado han sido Jupyter Notebook y Nextflow.

En este capítulo, se plasma cómo desarrollar un primer flujo de trabajo del proyecto de MEvoLib en Jupyter Notebook, configurando la herramienta y construyendo un cuaderno de trabajo sencillo disponible para su ejecución.

En segundo lugar, se realiza una migración del workflow desarrollado en Jupyter Notebook a la herramienta de Nextflow para, posteriormente, añadir los módulos restantes y expandir la parametrización que permita una mayor configuración del flujo de trabajo.

4.1 Workflow en Jupyter Notebook

Tal y como se ha presentado en la instalación de las herramientas del capítulo SetDev, Jupyter Notebook permite la ejecución de código de varios lenguajes de programación con diversas formas de visualización.

El objetivo primero de este cuaderno del flujo de trabajo es recuperar todas las secuencias de ADN mitocondrial de referencia (RefSeq) en GenBank para el género de garrapatas Ixodes. Además, se analizan las secuencias descargadas para obtener estadísticas básicas sobre el proceso de ejecución, como la longitud media, máxima y mínima de las secuencias, cuáles no poseen información adicional y datos relevantes que puedan ser de interés para el desarrollo de futuras etapas del flujo.

En primer lugar, añadimos la etapa de descarga o fetching al flujo de trabajo de la siguiente manera:

Figura 8: Código de la fase de fetching, flujo de trabajo en Jupyter Notebook

```
from mevolib.fetch.BioSeqs import BioSeqs
```

```
seq_db = BioSeqs.from_entrez(email="ex@ucm.es", entrez_db="nuccore",
                             query='("Ixodes"[Organism] OR ixodes[All Fields]) AND (biomol_genomic[PROP] AND refseq[filter])'
                              max_fetch=10)
```

```
seq_db.statistics()
```

```
seq_db.write('mtDNA_Ixodes.gb')
```

Fuente: Elaboración propia

Y se analizan las secuencias descargadas para obtener estadísticas básicas relativas al número total de secuencias almacenadas, la media aritmética de los elementos de la matriz de las secuencias, la desviación estándar de la longitud de las secuencias y los valores extremos de su longitud.

Por último, se procede a guardar la información en un fichero con formato GENBANK con los objetos SeqRecord obtenidos.

De este modo, una vez obtenidas las secuencias deseadas, se procede a añadir la etapa de agrupamiento o clustering al flujo de trabajo, para realizar la división de las secuencias en genes.

Figura 9: Código de la fase de clustering, flujo de trabajo en Jupyter Notebook

```
from mevolib import cluster
```

```
gene_dict = cluster.get_subsets('genes','ixodes.gb','gb', None, None, None, 'ixodes.log')
```

Fuente: Elaboración propia

De este modo, con la ejecución secuencial de estos fragmentos de código, podemos formalizar un Jupyter Notebook en el que, automáticamente, se descarguen, analicen y agrupen las secuencias deseadas por el usuario.

Una vez ejemplificado este proceso, se procede a realizar la migración a Nextflow, herramienta con la que generalizaremos y parametrizaremos el flujo de trabajo presentado en Jupyter Notebook mediante la integración de los tres módulos principales de MEvoLib: Fetch, Cluster y Align.

4.2 Workflow en Nextflow

La segunda herramienta contemplada para la implementación del flujo de trabajo de MEvoLib es Nextflow. En este apartado, se continua con el objetivo de la sección anterior y se migra el flujo de trabajo desarrollado en Jupyter Notebook. A mayores, se extiende la parametrización para incluir tanto la personalización de dicho flujo como el módulo de Align, o alineamiento, de la biblioteca.

Como se menciona en capítulos anteriores, Nextflow presenta un marco de flujo de trabajo y lenguaje de programación DSL que facilita la escritura de flujos de trabajo computacionales intensivos de datos.

El script de flujo de trabajo implementado para el workflow está compuesto principalmente de tres procesos (*Processes*) diferentes, uno para cada módulo de Fetch, Cluster y Align, una llamada al flujo de trabajo (workflow) y un archivo de configuración.

En cuanto a los procesos, estos se ejecutan de manera independiente y están aislados entre sí, es decir, no comparten un estado común. Para la comunicación de los procesos se utilizan canales (*Channels*), tanto de entrada (`input`) como de salida (`output`). Estos canales son colas FIFO asíncronas.

Dentro de cada proceso, se observan tres bloques distintos. El primer bloque, `input`, define el canal de entrada por el que cada proceso recibe la información. Así mismo, el bloque `output` hace referencia al canal de salida del proceso. El tercer y último bloque es el `script`, que representa el comando que ejecuta el flujo.

Además, se especifica un directorio de publicación (`publishDir`) donde se almacenan los archivos de salida, con el modo de copia (`mode: 'copy'`) y la opción de no sobrescribir archivos existentes (`overwrite: false`).

En este caso, los scripts de cada proceso ejecutan un comando de Shell que realiza una llamado a las funciones `main` respectivas de cada módulo.

Dentro de las funciones `main` se leen los parámetros de entrada y se llama a la función principal de cada módulo.

La definición de estos comandos se realiza en el archivo `pyproject.toml` que se encuentra dentro de la biblioteca MEvoLib.

Cabe resaltar que los archivos del workflow con extensión de Nextflow (`.nf`) se desarrollan en el repositorio de `parasitology`, dentro de la rama `nextflow`, en la cual la llamada al flujo de trabajo y la configuración se encuentran en la carpeta `worflows` y los procesos en la carpeta `modules`.

En referencia al proceso del primer módulo, `Fetch`, se crea el archivo `fetch_seqs.nf`. En él, se incluye el primero de los procesos, llamado `FETCH_SEQS`.

Figura 10: Proceso del módulo de Fetch

```
process FETCH_SEQS {

    publishDir "./data/${name}/fetch", mode: 'copy', overwrite: false

    input:
    val total_query
    val name

    output:
    path "${name}.gb"

    shell:
    """
    fetch_genbank_seqs -q "$total_query" -o "$name"
    """
}
```

Fuente: Elaboración propia

Como se puede observar, se realiza una llamada a la función `main` de `Fetch` en `BioSeqs.py` para realizar la descarga de secuencias de la base de datos GenBank a partir de la consulta de búsqueda `total_query` y guardarla en un archivo de extensión .gb.

La función de `main` en `BioSeqs.py` llama, a su vez, a la función `call_fetch_gb_seqs`, la cual tiene la misma lógica que el flujo de trabajo implementado en Jupyter Notebook.

Figura 11: Función main de fetch/BioSeqs.py

```python
def call_fetch_gb_seqs(query: str, name: str) -> None:
    """Default call for from_entrez function"""
    seq_db = BioSeqs.from_entrez(
        email="user@example.com", entrez_db="nuccore", query=query)
    print(seq_db.statistics())
    seq_db.write(name + '.gb')


def main ():
    """Default call for BioSeqs module."""
    parser = argparse.ArgumentParser()
    parser.add_argument("-q", "--query", required=True, help="Query sentence (Entrez format)")
    parser.add_argument("-o", "--output", required=True, help="Output file name (without extension)")
    args = parser.parse_args()
    call_fetch_gb_seqs(args.query, args.output)
```

Fuente: Elaboración propia

En el fichero de `get_genes.nf`, se encuentra el proceso `GET_GENES` correspondiente al módulo `Cluster`.

Figura 12: Proceso del módulo Cluster

```
process GET_GENES {

   publishDir "./data/${name}/cluster", mode: 'copy', overwrite: false

   input:
   path genes
   val name

   output:
   path '*.fasta'

   shell:
   """
   get_genes -i "$genes" -o "$name"
   """
}
```

Fuente: Elaboración propia

El propósito de este proceso es realizar la llamada a la función `main` de `cluster/_init_.py` para ejecutar la división de las secuencias del fichero de entrada, descargado en el proceso anterior, en genes.

La función `main` de `Cluster` incluye una nueva funcionalidad en la biblioteca de MEvoLib. El método `get_subsets` en Cluster devuelve un diccionario con una clave por conjunto y una lista de objetos `SeqRecord` como valor asociado a cada clave. Para facilitar la inclusión en el pipeline, se opta por volcar cada secuencia (clave y valor) en un archivo de salida de formato FASTA.

Figura 13: Función main de cluster/_init_.py

```
def main():
    """Default call for Genes module."""
    parser = argparse.ArgumentParser(
        description="Performs the division of every sequence stored in the input file into subsets by genes"
    )
    parser.add_argument("-i", "--input", required=True, help="Input sequences file")
    parser.add_argument("-f", "--format", default="genbank", help="Input file format")
    parser.add_argument("-o", "--output", required=True, help="Output file name (without extension)")
    args = parser.parse_args()
    gene_dict = get_subsets('genes', args.input, args.format, log_file=args.output + '.log')

    #  We dump the split sequences stored in the dictionary into a fasta file.
    args_iterator = ((value, f"{key}.fasta") for key, value in gene_dict.items() if len(value)>0)
    pool = multiprocessing.Pool(processes=NUMCORES)
    pool.imap_unordered(parallel_seqio_write, args_iterator)
```

Fuente: Elaboración propia

En la última parte de esta función `main`, se utiliza la biblioteca multiprocessing para escribir cada subconjunto de genes del diccionario en paralelo. El número de procesos paralelos se define en una variable llamada `NUMCORES`.

Finalmente, se incluye el flujo de trabajo del módulo Align que se encuentra en el archivo `get_align.nf`. Para ello, se implementa el proceso `GET_ALIGN.`

Figura 14: Función main de cluster/_init_.py

```
process GET_ALIGN {

    publishDir "./data/${name}/align", mode: 'copy', overwrite: false

    input:
    path fasta
    val name

    output:
    path '*_align.fasta'

    shell:
    """
    get_align -t "${params.tool}" -i "$fasta" -o "$name"
    """
}
```

Fuente: Elaboración propia

Como se puede observar, este proceso se ejecuta para cada uno de los ficheros FASTA obtenidos en el proceso anterior, `GET_GENES`. Este proceso llama a la función `main` de `align/_init_.py` para alinear las secuencias del fichero utilizando la herramienta de alineamiento dada. En este caso, se utiliza por defecto la herramienta de alineamiento Mafft, explicada en capítulos anteriores.

La función `main` de `Align` lee los parámetros de entrada y llama a la función `get_alignment`, que devuelve un fichero FASTA por cada fichero de entrada con las secuencias alineadas. Los ficheros de salida de este proceso se guardan en el directorio de salida que se crea `"./data/${name}/align"`.

Figura 15: Función main de align/_init_.py

```
def main():
    """Default call for Align module."""
    parser = argparse.ArgumentParser(
        description="Performs the sequences alignment using the given alignment tool and arguments"
    )
    parser.add_argument("-t", "--tool", required=True, help="Alignment tool")
    parser.add_argument("-i", "--input", required=True, help="FASTA file of unaligned sequences")
    parser.add_argument("-o", "--output", required=True, help="Output file name (without extension)")
    args = parser.parse_args()

    # We split the input file to obtain its name
    filename = Path(args.input).stem
    get_alignment(binary=args.tool, infile=args.input, infile_format='fasta', args='default',
                  outfile= f"{filename}_align.fasta", outfile_format='fasta')
```

Fuente: Elaboración propia

Cabe destacar que, para todos los archivos de Nextflow que se encuentran en la carpeta de `modules` en el repositorio de `parasitology` y que contienen el proceso de cada módulo, se ha incluido una llamada workflow a su proceso. Esto se implementa con el objetivo de proporcionar la posibilidad de ejecutar el flujo de trabajo de cada proceso de forma individual y no en su conjunto. Para ello, se incluye otro archivo `nextflow.config` para definir las variables de configuración globales y los valores por defecto.

Finalmente, se crea el archivo `execute_nextflow.nf` para realizar la llamada al flujo de trabajo completo. Este documento se encuentra en la carpeta de `workflows` en el repositorio de `parasitology`. En dicha carpeta, también se encuentra el otro archivo de `nextflow.config` que define las variables del flujo de trabajo global.

Figura 16: Archivo nextflow.config

```
params {
    query = null
    name = null

    species = null
    seq_type = null
    ref_seq = null

    genes = null

    tool = 'mafft'
    file = null
}
```

Fuente: Elaboración propia

En el fichero de `execute_nextflow.nf`, además de realizar las llamas a los tres procesos, realiza una serie de comprobaciones de los parámetros introducidos por consola.

Entre estos parámetros se encuentran:

- name: el nombre del fichero de salida del proceso del módulo de Fetch, así como el nombre del directorio donde se guardan los ficheros de salida de cada proceso.
- query: consulta completa de búsqueda para la descarga de información en GenBank. Para facilitar la parametrización del flujo de trabajo, la variable query puede dividirse en tres variables (species, seq_type y ref_seq).
- species: variable para establecer la especie como única consulta de búsqueda.
- seq_type: variable opcional que permite seleccionar el tipo de secuencia (ej. mtDNA).
- ref_seq: variable opcional que permite seleccionar la referencia de la secuencia (ej. Mitochondrial).

Cabe destacar que, para consulta de búsqueda en este flujo de trabajo, se ha descartado el término "[All Fields]", para garantizar que las secuencias obtenidas se correspondan fielmente con el organismo deseado y, por tanto, no se realice la extracción de todas las secuencias que mencionen el término sin realmente corresponder a dicha especie.

Figura 17: Código de execute_nextflow.nf

```
// Import modules/subworkflows
include { FETCH_SEQS } from '../modules/fetch_seqs.nf'
include { GET_GENES } from '../modules/get_genes.nf'
include { GET_ALIGN } from '../modules/get_align.nf'

//Name of the output file
if (params.name == null) {
    print("Please, insert the file name")
    exit(1)
}

if (params.query == null) {
    if (params.species == null) {
        print("Please, insert either the query or the species")
        exit(1)
    } else {
        total_query = "\"${params.species}\"[Organism]"
        // Add the sequence type of the query (if provided)
        if (params.seq_type != null) {
            total_query += " AND ${params.seq_type}[PROP]"
        }
        // Add the reference sequence of the query (if provided)
        if (params.ref_seq != null) {
            total_query += " AND ${params.ref_seq}[filter]"
        }
    }
} else {
    if (params.species == null) {
        total_query = params.query
    } else {
        print("Please, insert either the query or the species, but not both")
        exit(1)
    }
}

workflow {
    FETCH_SEQS(total_query, params.name)
    files = GET_GENES(FETCH_SEQS.out, params.name).flatten()
    GET_ALIGN(files, params.name)
}
```

Fuente: Elaboración propia

Capítulo 5 - Código Fuente

Esta sección del presente documento pretende recoger brevemente las referencias a los repositorios y códigos fuentes utilizados para la elaboración del proyecto.

En primer lugar, el código fuente de la biblioteca MEvoLib se encuentra en el repositorio de GitHub de Jorge Álvarez Jarreta (JAlvarezJarreta/*MEvoLib: Molecular Evolution Library for Python*, 2016/2023).

https://github.com/JAlvarezJarreta/MEvoLib/tree/version/2.0

Este repositorio cuenta con las ramas main, dev/nextflow, dev/v2.0 y version/2.0. Esta última siendo sobre la que se ha trabajado principalmente.

Para trabajar en MEvoLib, se han creado dos repositorios a mayores haciendo una copia (fork) del repositorio original. Estos repositorios son:

(*sansanpre/MEvoLib: Molecular Evolution Library for Python*2022/2023)
https://github.com/sansanpre/MEvoLib/tree/version/2.0

(repmess/*MEvoLib: Molecular Evolution Library for Python*, 2022/2023)
https://github.com/repmess/MEvoLib/tree/version/2.0

A su vez, se ha usado el código fuente del repositorio de parasitology de Jorge Álvarez Jarreta (JAlvarezJarreta/parasitology, 2018/2022).

Este repositorio cuenta con las ramas de main, jupyterwf y nextflow. En la rama de jupyterwf se incluye el flujo de trabajo implementado en la herramienta de Jupyter Notebook y en la rama de nextflow se incorpora el flujo de trabajo generado con la herramienta de Nextflow.

Al igual que con el repositorio de MEvoLib, se ha creado un repositorio nuevo a través de la opción de copia (fork). Este repositorio es:

(sansanpre/parasitology, 2023/2022)
https://github.com/sansanpre/parasitology/tree/nextflow

Capítulo 6 - Caso de Uso

En el capítulo 1 del presente documento se explica la importancia del procesamiento y análisis de ADN, específicamente del ADN mitocondrial.

Es por ello que el caso de uso que se presenta en este apartado pretende mostrar el paso a paso para realizar la recopilación y el tratamiento de datos de secuencias de ADN mitocondrial de la familia de las garrapatas para su análisis.

La ejecución de la biblioteca y de su flujo de trabajo se realiza en la plataforma de Visual Studio Code utilizando la herramienta de Nextflow. Para ello, se debe tener el entorno virtual activo y encontrarse en el directorio raíz de MEvoLib y parasitology.

Para la query de consulta, se utilizan las variables species y ref_seq, de tal forma que:

- species = Dermacentor
- ref_seq = mitochondrion

Además, se propone como valor para la variable name dermacentor. La palabra Dermacentor hace referencia a un género de la familia de las garrapatas. Otro ejemplo de garrapata sería el género Ixodes.

Para obtener las estadísticas de la ejecución del flujo de trabajo con Nextflow, se incluye `-with-report` a la línea de comando cuando se lanza su ejecución.

Por lo tanto, la línea de comando final que se ejecuta es:

Figura 18: Ejemplo línea de comando para el caso de uso

```
(mtdna) sansan@MSI:~/TFG$ nextflow run parasitology/workflows/test_nextflow.nf --name dermacentor --species
Dermacentor --ref_seq mitochondrion -with-report ./data/dermacentor/reportDerma.html
```

Fuente: Elaboración propia

Tras la ejecución de la línea anterior, Nextflow lanza a ejecución los distintos procesos. Se puede observar la cantidad de veces que lanza cada proceso: en este caso 1 proceso ejecutado para Fetch y Cluster (que se ejecutan de forma secuencial), respectivamente, y 130 procesos ejecutados para Align (que se ejecutan de forma concurrente).

Figura 19: Ejemplo ejecución de Nextflow del caso de uso

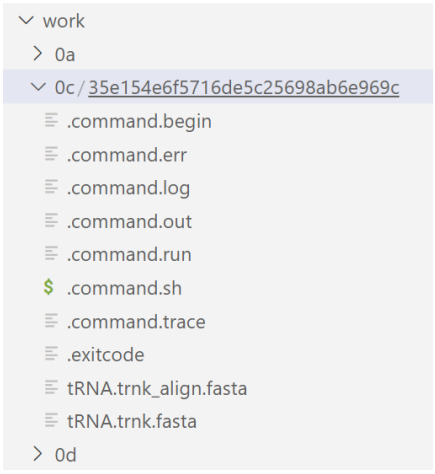
```
N E X T F L O W  ~  version 22.10.7
Launching `parasitology/workflows/test_nextflow.nf` [distracted_fermi] DSL2 - revision: e246f103f1
executor >  local (132)
[87/74c948] process > FETCH_SEQS      [100%] 1 of 1 ✓
[d8/508fb5] process > GET_GENES       [100%] 1 of 1 ✓
[54/f60bc9] process > GET_ALIGN (108) [100%] 130 of 130 ✓
Completed at: 13-May-2023 19:11:25
Duration    : 4m 52s
CPU hours   : 0.5
Succeeded   : 132
```

Fuente: Elaboración propia

Cuando finaliza el flujo de trabajo, aparecen dos directorios nuevos:

- work: en este directorio se almacenan los ficheros de salida de los procesos, así como los ficheros tipo command.out, command.log, command.err, command.begin, command.run y command.sh.

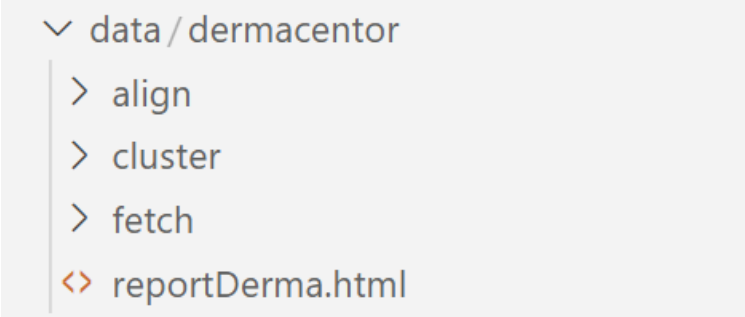
Figura 20: Ejemplo de directorio work del caso de uso



Fuente: Elaboración propia

- data: en este directorio se guardan los ficheros de salidas de los distintos procesos.

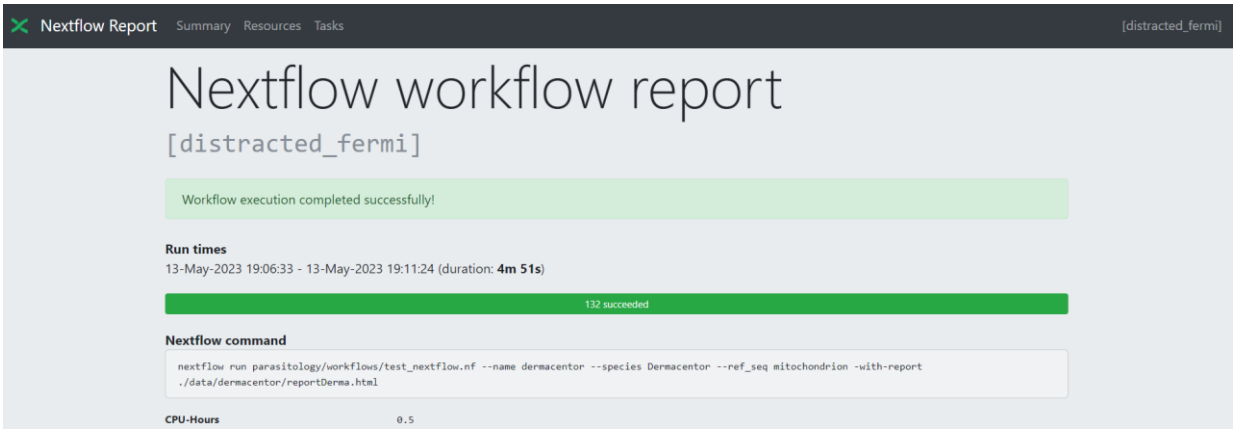
Figura 21: Ejemplo del directorio data del caso de uso



Fuente: Elaboración propia

Finalmente, en el directorio de data en la carpeta del ejemplo del caso de uso se encuentra el reporte generado por Nextflow, el cual es un archivo en formato HTML que se encuentra adjuntado en el Apéndice C.

Figura 22: Reporte generado en el caso de uso de la máquina local



Fuente: Elaboración propia

Además, con el objetivo de analizar el alcance del proyecto y su desempeño, se ha experimentado con la ejecución de MEvoLib y Nextflow en un servidor proporcionado por la Facultad de Informática de la Universidad Complutense de Madrid. De este modo, lanzando el proyecto en dicho servidor y observando el reporte correspondiente generado (ver Apéndice D) se puede apreciar que el tiempo de

ejecución de la misma query disminuye respecto al proyecto lanzado en la máquina local.

Figura 23: Información sobre la CPU del servidor

```
gporto@simba:/srv/home/gporto/TFG$ lscpu
Arquitectura:                         x86_64
modo(s) de operación de las CPUs:     32-bit, 64-bit
Orden de los bytes:                   Little Endian
Tamaños de las direcciones:           46 bits physical, 48 bits virtual
CPU(s):                               12
Lista de la(s) CPU(s) en línea:       0-11
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»:               6
«Socket(s)»                           1
Modo(s) NUMA:                         1
ID de fabricante:                     GenuineIntel
Familia de CPU:                       6
Modelo:                               85
Nombre del modelo:                    Intel(R) Xeon(R) W-2235 CPU @ 3.80GHz
Revisión:                             7
CPU MHz:                              4144.911
CPU MHz máx.:                         4600,0000
CPU MHz mín.:                         1200,0000
BogoMIPS:                             7599.80
Virtualización:                       VT-x
Caché L1d:                            192 KiB
Caché L1i:                            192 KiB
Caché L2:                             6 MiB
Caché L3:                             8,3 MiB
```

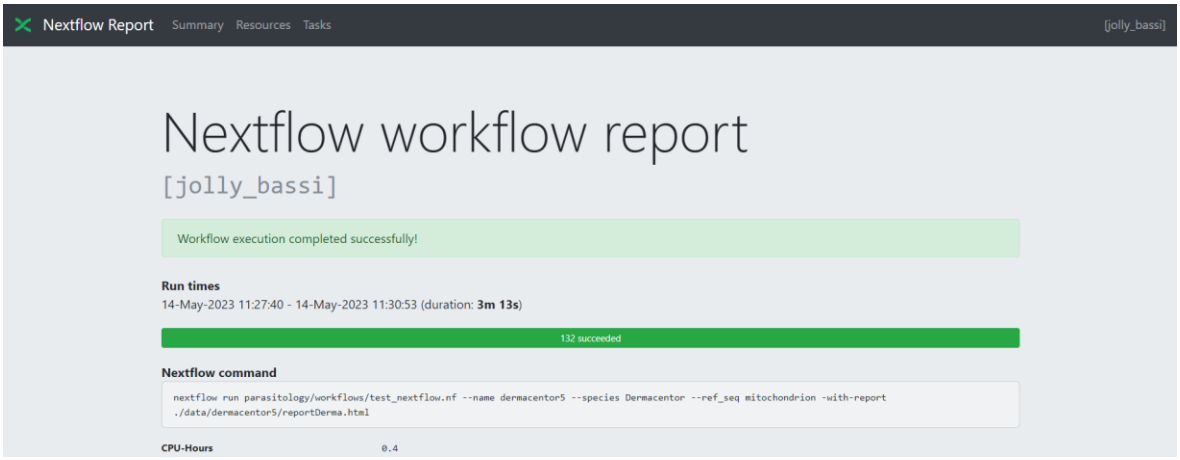
Fuente: Elaboración propia

Figura 24: Información sobre la CPU de la máquina local

```
sansan@MSI:~$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU(s):                          8
On-line CPU(s) list:             0-7
Thread(s) per core:              2
Core(s) per socket:              4
Socket(s):                       1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           140
Model name:                      11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz
Stepping:                        1
CPU MHz:                         2995.202
BogoMIPS:                        5990.40
Virtualization:                  VT-x
Hypervisor vendor:               Microsoft
Virtualization type:             full
L1d cache:                       192 KiB
L1i cache:                       128 KiB
L2 cache:                        5 MiB
L3 cache:                        12 MiB
```

Fuente: Elaboración propia

Figura 25: Reporte generado en el caso de uso del servidor



Fuente: Elaboración propia

Asimismo, con la comparación de ambos reportes (ver Apéndices C y D) se puede apreciar que el uso de máquinas más potentes, como es el caso del servidor, reducirá hasta un 33% el uso de CPU, escalable a proyectos de mayor volumen. Además del servidor, el flujo de trabajo de Netxflow se puede ejecutar en servidores en la nube, que mejorarían el rendimiento aún más.

De esta manera, se puede apreciar claramente que el módulo de Fetch supone un cuello de botella en cualquiera de las máquinas, ya que las conexiones de este método han de realizarse a la base de datos de GenBank y, por tanto, dependen del tráfico que haya en esta y de la velocidad máxima que permita la conexión.

Por otro lado, la ventaja significativa recae en el uso de recursos más potentes en el método de Align, en el que se pueden realizar los alineamientos, en este caso de las 132 secuencias, en paralelo para así disminuir notablemente la métrica relativa al consumo de CPU.

Capítulo 7 - Conclusiones y Trabajo Futuro

Tal y como se indica en el Capítulo 1 destinado a la introducción, las motivaciones de este trabajo se plasman en la consecución de diversos objetivos manifestados, también, en dicho capítulo.

De este modo, para concluir el trabajo realizado en este proyecto, se procede a evaluar la realización de dichos objetivos, las limitaciones que se han presentado y las futuras acciones propuestas.

En primer lugar, la configuración de la infraestructura de desarrollo se ha completado correctamente, ya que sobre ella se ha procedido a completar la actualización de la biblioteca MEvoLib, terminada en su totalidad y presentando, así, la biblioteca adaptada totalmente a la versión de Python 3.10.

De este modo, con el desarrollo del flujo de trabajo en Jupyter Notebook y posteriormente en Nextflow, se consigue presentar un flujo de trabajo organizado, sencillo, útil y eficaz con el que el usuario puede desarrollar sus trabajos con la obtención, limpieza y alineación de las secuencias de ADN que considere necesarias para sus investigaciones.

Además, con la escritura del presente documento, el usuario podrá reforzar cualquier concepto que considere pertinente, además de poder consultar cualquier duda en cuanto al desarrollo del código fuente, la descripción precisa de los módulos de la biblioteca y el ejemplo del caso de uso que puede guiarle para el desarrollo de su propia investigación.

El flujo de trabajo implementado para la ejemplificación de varios casos de uso ha sido validado por un experto en el ámbito de las ciencias biológicas, en el que la utilidad de MEvoLib se vio reflejada tanto en eficacia como en interés para el desarrollo de futuros trabajos de investigación.

Por otro lado, una de las limitaciones principales que se ha identificado durante el desarrollo del proyecto ha sido el gran volumen de mejoras y novedades que se

podrían desarrollar entorno al objeto de estudio del presente trabajo, que quedan fuera del alcance de este debido a que no el plan de trabajo no pudo contemplarlas por falta de, principalmente, tiempo y carga de trabajo. Como futura mejora consideramos, por ejemplo, la configuración e implementación de pruebas software mediante la biblioteca pytest; o la actualización del módulo de MEvoLib encargado en la construcción de árboles filogenéticos, formado por los módulos de inference y phyloassemble que todavía se encuentran en la versión original de Python, así como su incorporación como nueva tarea al flujo de trabajo diseñado en Nextflow.

Por último, de cara al trabajo futuro, además de volver a considerar las limitaciones mencionadas, encontramos relevancia en presentar una interfaz de usuario más vistosa y amigable, en la que la parametrización se pueda realizar por medio de menús y botones disponibles en la aplicación, mejorando así la experiencia de usuarios menos familiarizados con la ejecución de comandos en terminales. Además, se valora la opción de ampliar los casos de uso a diferentes ámbitos científicos, expandiendo la biblioteca a otros sectores de la biología, en los cuales podrá impulsar y facilitar trabajos de investigación.

Chapter 7' - Conclusion and Future Work

As indicated in Chapter 1, dedicated to the introduction, the motivations of this work are reflected in the achievement of various objectives stated, also, in said chapter.

Therefore, to conclude the work carried out in this project, we proceed to evaluate the achievement of these objectives, the limitations that have been presented and the future actions proposed.

Firstly, the configuration of the development infrastructure has been completed correctly, since the update of the MEvoLib library has been completed on it, finished in its entirety and presenting, thus, the library fully adapted to the version of Python 3.10.

With the development of the workflow in Jupyter Notebook and, mainly, in Nextflow, it is possible to present an organized, simple, useful and efficient workflow with which the user can carry out their work with the obtaining, cleaning and alignment of DNA sequences that you deem necessary for your research.

Moreover, with the writing of this document, the user will be able to reinforce any concept that he considers pertinent, in addition to being able to consult any doubt regarding the development of the source code, the precise description of the library modules and the example of the use case that can guide you in the development of your own research.

Additionally, the workflow implemented for the exemplification of several use cases has been validated by an expert in the field of biological sciences, in which the utility of MEvoLib was reflected both in efficiency and in interest for the development of future Investigation work.

On the other hand, one of the main limitations that has been identified during the development of the project has been the large volume of improvements and novelties that could be developed around the object of study of this work, which are outside the scope of this since the work plan could not contemplate them due to lack of, mainly, time and workload. As a future improvement, it is consider, as an example, the

configuration and implementation of software tests using pytest library; or the update of the MEvoLib module in charge of the construction of phylogenetic trees, formed by the inference and phyloassemble modules that are still int the original version of Python, as well as its incorporation to the Nextflow workflow.

such as, for example, the implementation of the files corresponding to the part of the test framework pytest.

Finally, regarding future work, in addition to reconsidering the aforementioned limitations, it is relevant to present a more visually appealing user interface that allows parameterization through menus and buttons available in the application, thus improving the experience of users less familiar with executing commands in terminals. Moreover, expanding the use cases to different scientific fields is considered valuable, expanding the library to other sectors of biology, in which it can promote and facilitate research work.

Capítulo 8 - Contribuciones Personales

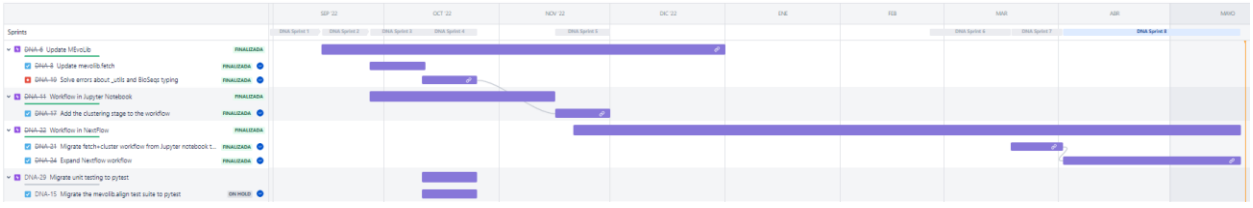
La elaboración del presente trabajo se ha realizado de manera conjunta y en un ambiente de colaboración óptimo y eficaz. A pesar de encontrar división de tareas en varias fases, la cooperación ha sido absoluta ante la resolución de problemas e implementación de mejoras.

Las tareas individuales se presentan en los siguientes apartados, teniendo en cuenta que el resto de tareas han sido realizadas en conjunto y el desglose detallado se puede observar en la Figura 1 del apartado sobre la gestión de tareas.

8.1 Gloria Porto Cabero

- ☑ Configuración de la infraestructura SetDev.
- ☑ Actualización de MEvoLib: módulo Fetch.
- ☑ Flujo de trabajo en Jupyter Notebook: módulo Cluster.
- ☑ Flujo de trabajo en Nextflow: migración del módulo Fetch.
- ☑ Desarrollo de la memoria de trabajo

Figura 26: Diagrama de las contribuciones personales de Gloria Porto Cabero

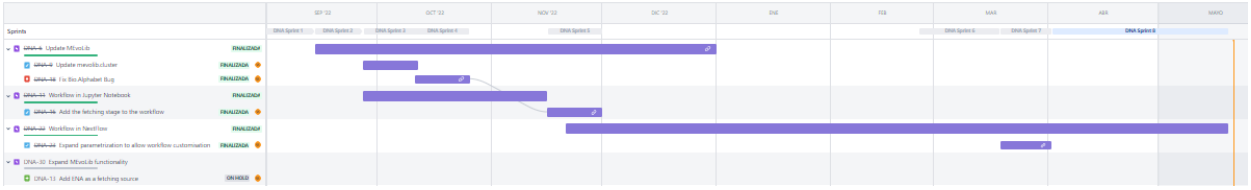


Fuente: Herramienta Jira (Atlassian, 2023).

8.2 Sandra Sánchez Prevost

- ☑ Configuración de la infraestructura SetDev.
- ☑ Actualización de MEvoLib: módulo Cluster.
- ☑ Flujo de trabajo en Jupyter Notebook: módulo Fetch.
- ☑ Flujo de trabajo en Nextflow: migración del módulo Cluster.

Figura 27: Diagrama de las contribuciones personales de Sandra Sánchez Prevost



Fuente: Herramienta Jira (Atlassian, 2023).

Bibliografía

Atlassian, (2023). Herramientas de gestión ágil de proyectos para equipos de software. Atlassian. Recuperado 3 de abril de 2023, de https://www.atlassian.com/es/software/jira/agile

Alvarez-Jarreta, J. (2022). JAlvarezJarreta/MEvoLib [Python]. GitHub. https://github.com/JAlvarezJarreta/MEvoLib (Obra original publicada en 2016).

Álvarez Jarreta, J. (2017). Molecular phylogenetic analysis: design and implementation of scalable and reliable algorithms and verification of phylogenetic properties (Tesis doctoral). Recuperado el 23 de abril de 2023, de https://zaguan.unizar.es/record/61541/files/TESIS-2017-041.pdf

Álvarez-Jarreta, J. y Ruiz-Pesini, E. (2016). MEvoLib v1.0: The First Molecular Evolution Library for Python. BMC Bioinformatics, 17(436), 1-8.

Apache. (n.d.). Licenses. Recuperado el 14 de septiembre de 2022, de https://www.apache.org/licenses/

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., & Sayers, E. W. (2010). GenBank. Nucleic Acids Research, 38, 46-51.

Biopython Developers. (2020). Biopython 1.76. [Software]. Bio.Entrez module. Retrieved from https://biopython.org/docs/1.76/api/Bio.Entrez.html

Build software better, together. (s. f.). GitHub. Recuperado 12 de mayo de 2023, de https://github.com

Cock, P.J.A., Antao, T., Chang, J.T., Chapman, B.A., Cox, C.J., Dalke, A., ... de Hoon, M.J.L. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), 1422-1423.

Edgar, R.C. (2004). Muscle: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res., 32(5), 1792–7.

Felsenstein, J. (2006). Phylogeny inference package (PHYLIP). Recuperado el 23 de abril de 2023, de https://evolution.genetics.washington.edu/phylip/general.html.

Katoh, K. (2020). MAFFT - Ubuntu on Windows. Recuperado el 13 de abril de 2023, de https://mafft.cbrc.jp/alignment/software/ubuntu_on_windows.html

Katoh, K., Misawa, K., Kuma, K., & Miyata, T. (2002). MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Research, 30(14), 3059-3066

National Center for Biotechnology Information, NCBI. Recuperado el 10 de Abril de 2023, de https://www.ncbi.nlm.nih.gov/

Nelesen, S., Liu, K., Wang, L. S. K., Linder, C. R., & Warnow, T. (2012). DACTAL: divide-and-conquer trees (almost) without alignments. Bioinformatics, 28, 274-282.

Nextflow's documentation! —Nextflow 23.04.1 documentation. (s. f.). Recuperado 12 de mayo de 2023, de https://www.nextflow.io/docs/latest/index.html

Pyenv-virtualenv. (2023). [Shell]. pyenv. https://github.com/pyenv/pyenv-virtualenv (Original work published 2013)

Price, M. N., Dehal, P. S., & Arkin, A. P. (2009). FastTree: Computing Large Minimum-Evolution Trees with Profiles instead of a Distance Matrix. Molecular Biology and Evolution, 26, 1641–1650.

Sievers, F., Wilm, A., Dineen, D., Gibson, T., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., Thompson, J., & Higgins, D. (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Molecular Systems Biology, 7, 539.

Simple Python Version Management: Pyenv. (2023). [Roff]. pyenv. https://github.com/pyenv/pyenv (Original work published 2012)

Stamatakis, A. (2006). RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics, 22(21), 2688–2690.

Visual Studio Code—Code Editing. Redefined. (s. f.). Recuperado 13 de mayo de 2023, de https://code.visualstudio.com/

Welcome to Python.org. (2023, mayo 11). Python.Org. https://www.python.org/

Apéndices

Apéndice A – Glosario de términos

Con el objetivo de facilitar la lectura y entendimiento de este proyecto, en el presente apéndice se recogen los conceptos más relevantes a tener en cuenta en relación al objeto de estudio.

- Secuencia de ADN: conjunto lineal de nucleótidos (moléculas) que conforman la molécula compleja de ADN, en la cual se encuentra la información genética del organismo.

- Mitocondria: órgano celular encargado de la producción de energía para la célula eucariota.

- ADN nuclear: forma predominante del ADN en las células, organizado en forma de cromosomas y heredado de ambos progenitores. Contiene la mayor parte de la información genética de un organismo.

- ADN mitocondrial: molécula de ADN dentro de las mitocondrias, heredado únicamente de la madre y con la información genética necesaria para la producción de proteínas y enzimas.

- Filogenética: estudio de las relaciones evolutivas entre organismos y su historia evolutiva.
- Alineamiento de secuencias: técnica de bioinformática empleada para comparar dos o más secuencias biológicas y localizar las regiones en las que se asemejan.

- Big Data: conjunto de datos masivos, complejos y variables cuyo tratamiento implica técnicas de procesamiento especializadas.

- Entorno virtual: un entorno virtual de Python es una copia independiente del intérprete y las bibliotecas instaladas en un sistema que se puede utilizar para trabajar en proyectos sin que afecte a otros.

- Python: lenguaje de programación interpretado de alto nivel utilizado en multitud de sectores por su sencillez y claridad.

- Biopython: biblioteca de Python de utilidad en el ámbito de la bioinformática, diseñada para procesamiento, análisis y visualización de datos biológicos por parte de programadores y científicos.

- Domain Specific Languaje (DSL): el Lenguaje Específico de Dominio es una lenguaje de programación diseñao para abordar un ámbito específico de aplicación. A diferencia de los lenguajes de programación generales, los DSL se centran en un dominio particular, proporcionando una sintaxis y abstracciones específicas para ese ámbito.

Apéndice B – Manual de instalación

Como se ha comentado en el apartado de Plan de Trabajo, en este apéndice se explica el proceso de instalación que se ha seguido para configurar las herramientas y entornos de trabajo.

Todos los manuales y páginas oficiales mencionados en los siguientes párrafos se encuentran recogidos en la bibliografía, para facilitar el acceso y lectura del presente apéndice.

Antes de comenzar, conviene destinar un directorio exclusivo en nuestro sistema de ficheros de Linux. En este caso, utilizaremos como ejemplo el directorio ./TFG en Ubuntu.

Accediendo a la terminal, el siguiente paso es instalar Python. Como se va a trabajar con diferentes versiones de este lenguaje, ya que uno de los objetivos es actualizar MEvoLib a una versión más moderna, se ha optado por implementar la herramienta de gestión de versiones Pyenv, la cual permite administrar fácilmente múltiples versiones de Python en el mismo sistema. De esta forma, es posible instalar distintas versiones en el ordenador y cambiar entre ellas según sea necesario. A su vez, esta herramienta se encarga de configurar el entorno para cada versión de Python, incluyendo, por ejemplo, las bibliotecas estándar y la ruta de acceso a los binarios.

Para instalar Pyenv rápidamente, se puede seguir el manual que encontramos en la página oficial citada en la bibliografía, clonando el repositorio correspondiente y siguiendo la metodología descrita.

Una vez configurado Pyenv, se procede a instalar la versión de Python necesaria, en este caso, la versión 3.10.6.

Posteriormente, se procede a crear el entorno virtual en el que desarrollar el trabajo. Para ello, se instala la herramienta de Pyenv-Virtualenv, plugin de Pyenv que

permite gestionar y crear entornos virtuales de Python. Cada entorno virtual creado se almacena en una carpeta separada en el sistema, para así mantener versiones diferentes de Python y bibliotecas en cada proyecto.

A continuación, se crea y activa un entorno virtual al que, por ejemplo, denominar mtdna, con la versión de Python 3.10.6 asignada.

Una vez configurada e instalada la infraestructura general, se procede a utilizar la herramienta GitHub, plataforma de almacenaje y colaboración de código fuente que permite a los desarrolladores trabajar en proyectos en equipo. El uso de esta herramienta posibilita el guardado, revisión y colaboración en el código por parte de varios usuarios, además de contribuciones en proyectos de código abierto.

Asimismo, también conviene utilizar una plataforma de entorno de desarrollo integrado (IDE) como, por ejemplo, Visual Studio Code. Esta plataforma IDE de Microsoft ofrece una gran cantidad de herramientas y características como editores de código, depuradores, herramientas de control de versiones, compiladores o diseñadores de interfaces de usuarios; además de ser compatible con multitud de lenguajes de programación.

A continuación, se detalla la instalación de las herramientas necesaria para alcanzar los objetivos que se detallan en el presente trabajo.

Para la actualización de la biblioteca MEvoLib se realiza, en primera instancia, una ramificación del repositorio original de la biblioteca de Jorge Álvarez Jarreta (Alvarez-Jarreta, 2016/2022) a repositorios propios.

Una vez creado el repositorio de MEvoLib en la cuenta personal de GitHub, se debe clonar e instalar en el propio sistema, desde la consola de comandos. La clonación se realiza para la rama `version/2.0` sobre la que se trabaja.

De este modo, el entorno está preparado para actualizar la biblioteca a una versión más reciente de Python, como se detalla en el Capítulo 3.

Para implementar el flujo de trabajo de la biblioteca, explicado en el Capítulo 4, ha sido necesario configurar dos herramientas.

En primer lugar, se realiza un flujo de trabajo inicial en Jupyter-Notebooks. Esta herramienta es un entorno de programación interactivo basado en la web que permite crear y compartir documentos de código ejecutable, texto explícito y visualizaciones. Cabe destacar que su nombre, Jupyter, proviene de los lenguajes de programación que soporta (Julia, Python y R).

Para el segundo flujo de trabajo se utiliza la herramienta de Nextflow, una plataforma de flujo de trabajo diseñada para el análisis de datos y la bioinformática. En ella, se pueden definir y ejecutar flujos de trabajo de manera sencilla y reproducible en distintos entornos como clústeres, servidores en la nube y computación local. Además, es interesante mencionar que Nextflow utiliza un lenguaje de programación basado en DSL o Lenguaje Específico de Dominio y que, para su instalación, se ha de tener una versión de Java 8 o posterior.

Finalmente, es necesaria la instalación de la herramienta de Mafft para por realizar los alineamientos en el flujo de trabajo. Mafft es un software de alineación múltiple de secuencias de ADN, ARN y proteínas. De este modo, la biblioteca y, en concreto, el módulo de Align tienen a su disposición la herramienta de Mafft para ejecutar la alineación múltiple de secuencias, en este caso de ADN mitocondrial.

Apéndice C – Reporte de la máquina local

Nextflow workflow report

[distracted_fermi]

Workflow execution completed successfully!

Run times

13-May-2023 19:06:33 - 13-May-2023 19:11:24 (duration: **4m 51s**)

132 succeeded

Nextflow command

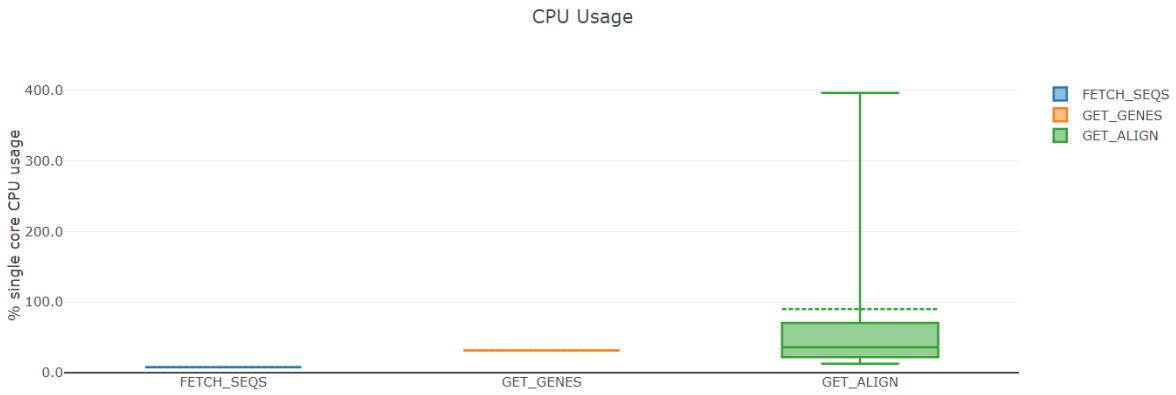
```
nextflow run parasitology/workflows/test_nextflow.nf --name dermacentor --species Dermacentor --ref_seq mitochondrion -with-report
./data/dermacentor/reportDerma.html
```

Resource Usage

These plots give an overview of the distribution of resource usage for each process.

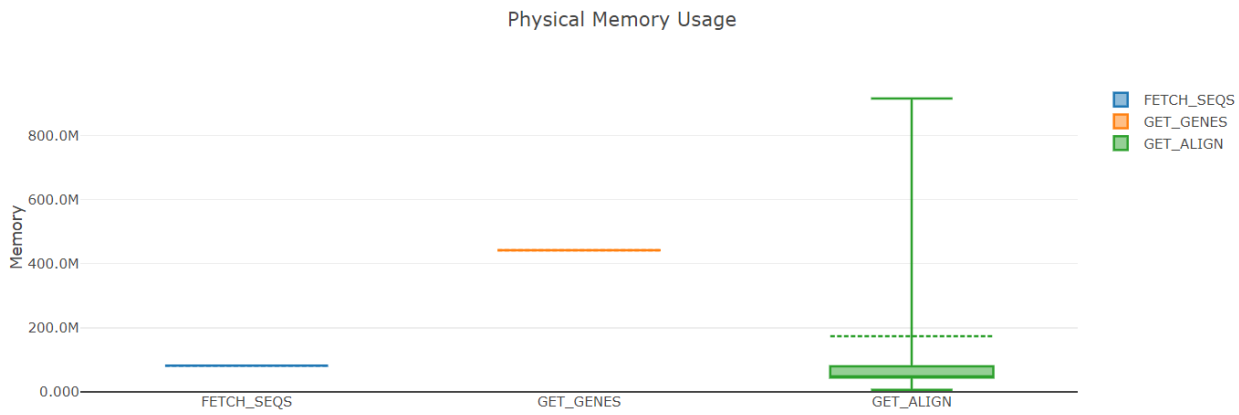
CPU

Raw Usage | % Allocated



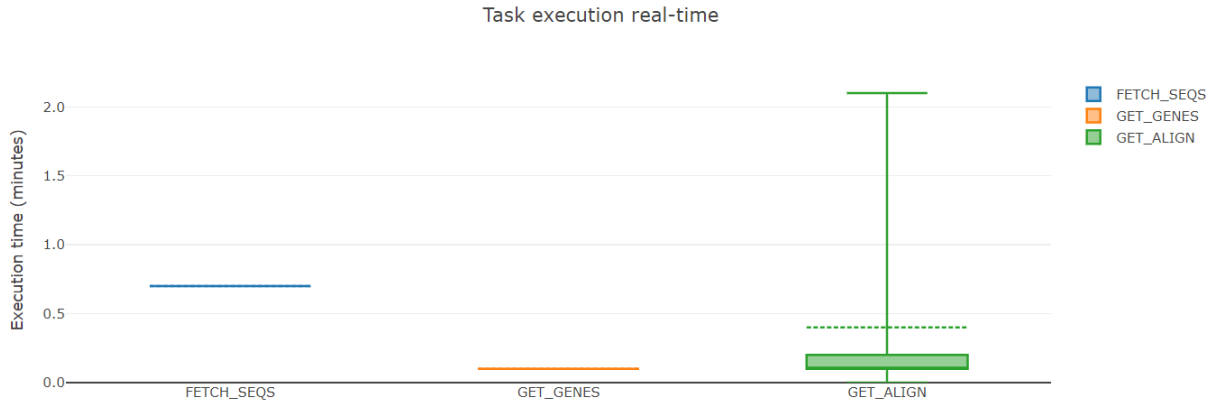
Memory

Physical (RAM) | Virtual (RAM + Disk swap) | % RAM Allocated



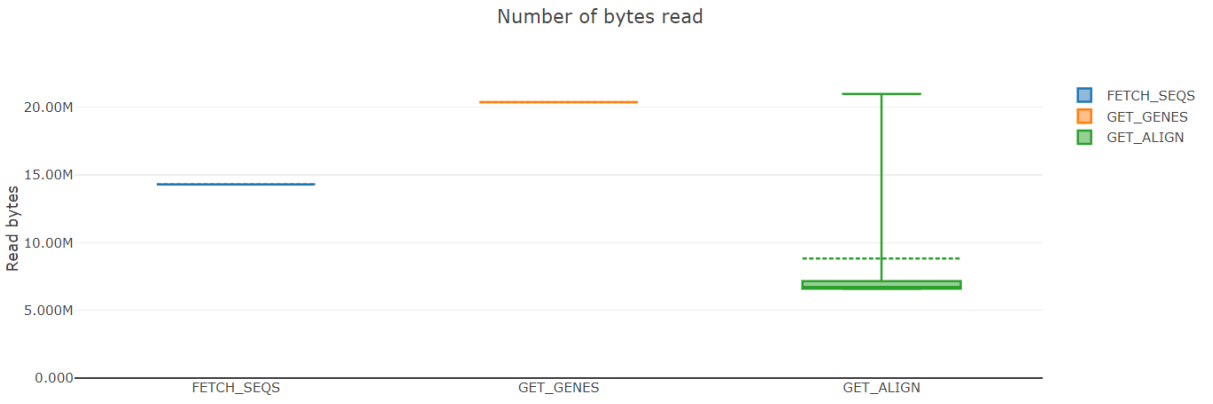
Job Duration

Raw Usage | % Allocated



I/O

Read | Write



Apéndice D – Reporte del servidor

Nextflow workflow report

[jolly_bassi]

Workflow execution completed successfully!

Run times

14-May-2023 11:27:40 - 14-May-2023 11:30:53 (duration: **3m 13s**)

132 succeeded

Nextflow command

```
nextflow run parasitology/workflows/test_nextflow.nf --name dermacentor5 --species Dermacentor --ref_seq mitochondrion -with-report
./data/dermacentor5/reportDerma.html
```

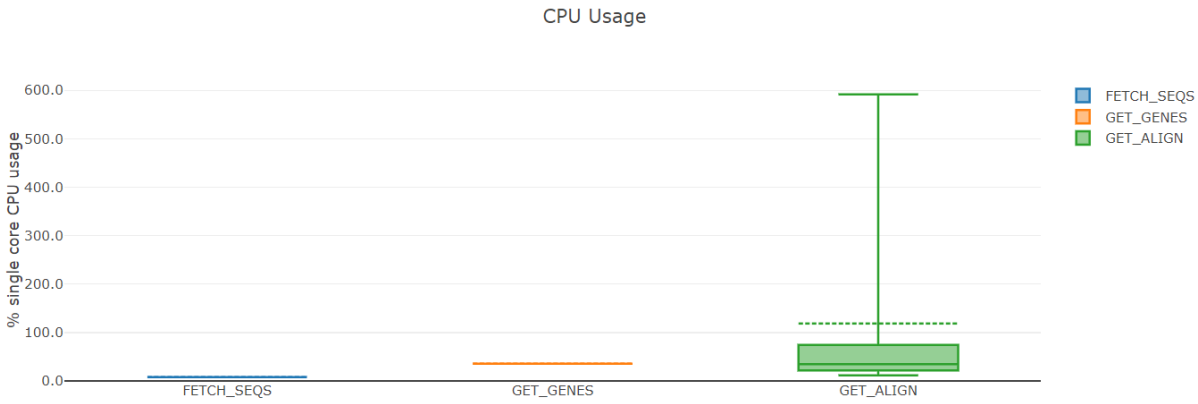
CPU-Hours	0.4

Resource Usage

These plots give an overview of the distribution of resource usage for each process.

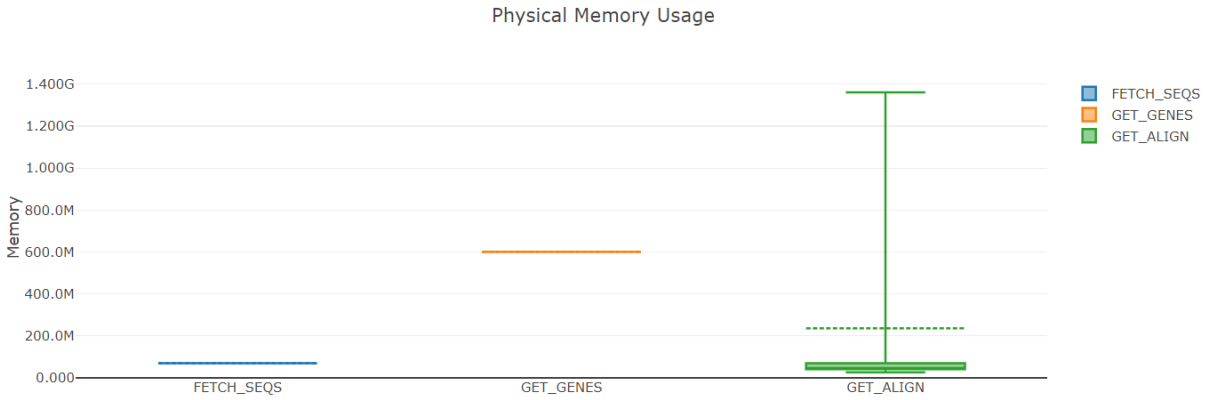
CPU

Raw Usage | % Allocated



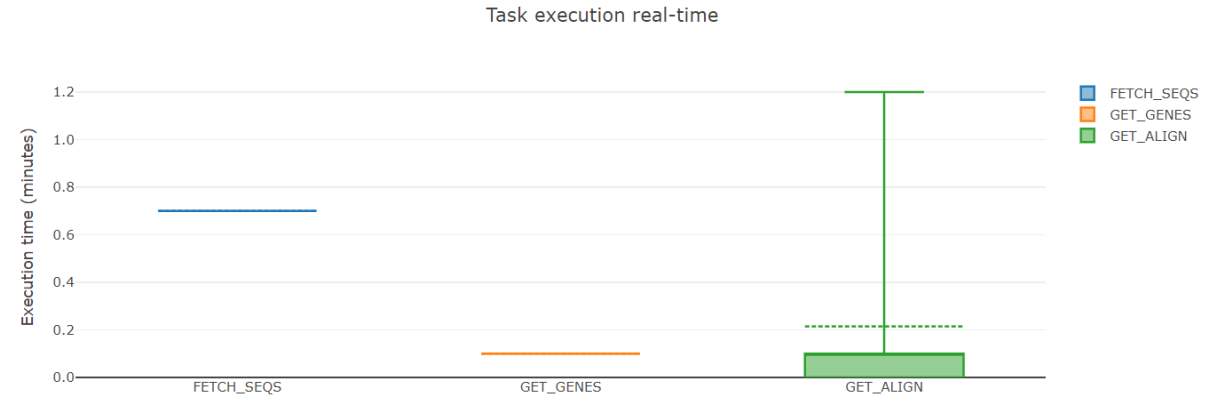
Memory

Physical (RAM) | Virtual (RAM + Disk swap) | % RAM Allocated



Job Duration

Raw Usage | % Allocated



I/O

Read | Write