16SPyPrimer: un paquete en Python para el análisis de la cobertura de *primers* del gen 16S rRNA

Lara María Vázquez González
Máster en Bioinformática y Bioestadística
Área 3

María José Carreira Nouche
Carlos Balsa Castro
Dorcas Orengo Ferriz

Junio 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*16SPyPrimer: un paquete en Python para el análisis de la cobertura de primers del gen 16S rRNA*
Nombre del autor:	*Lara María Vázquez González*
Nombre del consultor/a:	Dorcas J. Orengo Ferriz
Nombre del PRA:	Ferran Prados Carrasco
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	Máster en Bioinformática y Bioestadística
Área del Trabajo Final:	*Área 3*
Idioma del trabajo:	*Castellano*
Palabras clave	*Cobertura, primers, Python*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

En diversas áreas como microbiología, ecología o agricultura es importante y necesario identificar adecuadamente las especies bacterianas presentes en una muestra cualquiera. Un ejemplo puede ser asociar en un humano, animal o planta, una patología concreta por la presencia o no de una especie bacteriana concreta.

En muchos ámbitos de la investigación sobre la diversidad bacteriana es necesario tener la capacidad de identificar grupos concretos de bacterias y rechazar otros grupos para diferentes niveles taxonómicos. El gen 16S rRNA es de uso habitual desde hace muchos años como marcador filogenético para la identificación bacteriana, generalizándose su uso como gen diagnóstico para la identificación taxonómica.

En la literatura están descritos muchos pares de cebadores que funcionan adecuadamente para las diferentes zonas conservadas del gen 16S rRNA. Sin embargo, se suelen utilizar cebadores genéricos que no permiten seleccionar familias o géneros bacterianos de interés.

En este trabajo se desarrolla una herramienta en Python, llamada *16SPyPrimer,* que permite realizar un análisis de la cobertura de pares de cebadores para conjuntos específicos de bacterias. Este software permite a los investigadores modificar cebadores ya conocidos para encontrar otros más específicos según sus necesidades.

Esta herramienta posee amplias opciones configurables según el criterio del usuario y devuelve múltiples archivos de resultados con la información necesaria para realizar análisis propios *a posteriori*.

Abstract (in English, 250 words or less):

In various areas such as microbiology, ecology, or agriculture, it is important and necessary to properly identify the bacterial species present in any sample. An example may be the association of a specific pathology in a human, animal or plant due to the presence or absence of any specific bacterial species.

In many areas of research on bacterial diversity, it is necessary to be able to identify specific groups of bacteria and reject other groups for different taxonomic levels. The 16S rRNA gene has been used for many years as a phylogenetic marker for bacterial identification, being used as a diagnostic gene for taxonomic identification.

Many pairs of primers are described in the literature that work adequately for the different conserved areas of the 16S rRNA gene. However, generic primers that do not allow the selection of bacterial families of interest are often used.

The aim of this project is to develop a tool in Python, called 16SPyPrimer, that analyses the coverage of pairs of primers for specific sets of bacteria. This software will allow researchers to modify already known primers to find more specific ones according to their needs.

This tool has extensive configurable options according to the user's criteria and returns multiple results files with the necessary information to carry out other analysis afterwards.

Índice

1. Introducción 1
1.1 Contexto y justificación del trabajo 1
1.2 Objetivos del trabajo 3
1.2.1 Objetivos generales 3
1.2.2 Objetivos específicos 4
1.3 Enfoque y método seguido 5
1.4 Planificación del Trabajo 5
1.4.1 Tareas 5
1.4.1.1 Primera fase: desarrollo de la funcionalidad 6
1.4.1.2 Segunda fase: usabilidad, configurabilidad y pruebas complejas 7
1.4.1.3 Tercera fase: memoria, presentación y defensa 8
1.4.2 Calendario 8
1.4.3 Hitos 12
1.4.4 Análisis de riesgos 13
1.5 Productos obtenidos 14
1.6 Estructura de la memoria 14
2. Análisis 17
2.1 Análisis en el ámbito de la biología 17
2.2 Análisis de la funcionalidad 19
2.3 Análisis en el ámbito de la informática 21
3. Diseño 23
3.1 Estructura de clases 23
3.2 Método de búsqueda de cebadores 24
4. Implementación 26
4.1 Primera fase: desarrollo de la funcionalidad 26
4.1.1 Desarrollo de la funcionalidad básica 26
4.1.2 Ampliación de las funcionalidades 28
4.2 Segunda fase: usabilidad, configurabilidad y pruebas complejas 30
4.2.1 Mejoras de usabilidad 30
4.2.2 Mejoras de funcionalidad 31
5. Pruebas y reflexión. 35
5.1 Pruebas del software 35
5.2 Análisis de los resultados obtenidos 39
6. Conclusiones 41
6.1 Lecciones aprendidas 41
6.2 Objetivos 41
6.3 Planificación y metodología 42
6.4 Posibles mejoras 43
7. Glosario 44
8. Bibliografía 46
9. Anexos 49
9.1 Manual de usuario 49
9.2 Estructura de archivos de resultados 51

Lista de figuras

Ilustración 1: Estructura del ARN ribosómico 16S. Fuente: [3]..... 1
Ilustración 2: Diagrama de Gantt..... 11
Ilustración 3: Regiones del gen 16S. Fuente: [17]..... 18
Ilustración 4: Diagrama de clases definido para el desarrollo de 16SPyPrimer 24
Ilustración 5: Ejecución del comando *help* 49

1. Introducción

1.1 Contexto y justificación del trabajo

Los ácidos nucleicos y las proteínas son macromoléculas (polímeros) formadas por la unión de nucleótidos y aminoácidos (monómeros), respectivamente. Ambos polímeros son comunes a todos los seres vivos y cambian con el tiempo, por lo que pueden considerarse como relojes moleculares. Teniendo en cuenta que dichos cambios se producen al azar y aumentan con el tiempo de manera lineal, las diferencias en la secuencia de los monómeros que integran macromoléculas homólogas presentes en dos formas de vida reflejan la distancia evolutiva entre ellas. Este planteamiento se ha utilizado para establecer las relaciones filogenéticas o de parentesco entre los distintos seres vivos.

El ARN ribosómico (ARNr) 16S es la macromolécula más ampliamente utilizada en los estudios de filogenia y taxonomía bacterianas [1][2]. En la actualidad, la identificación de dicha macromolécula se lleva a cabo mediante la secuenciación del gen que la codifica.

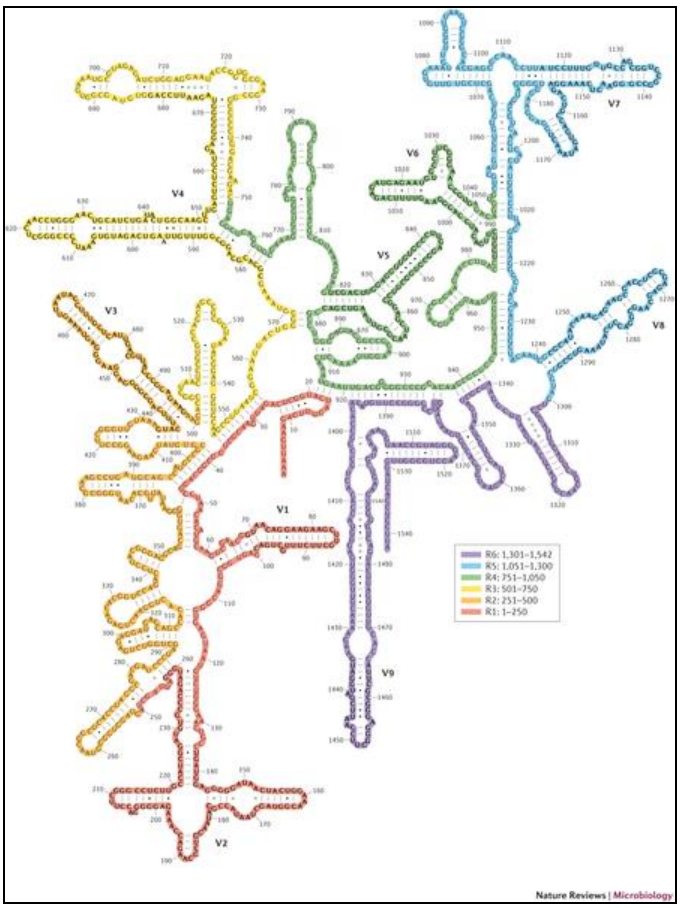

Ilustración 1: Estructura del ARN ribosómico 16S. Fuente: [3]

El ARNr 16S es un componente de la subunidad menor (30S) de los ribosomas procariotas que consta de aproximadamente 1500 nucleótidos. El gen que codifica para este ARN se denomina *rrs* o ADN ribosomal 16S (ADNr 16S) y a partir de su secuencia se puede obtener información filogenética y taxonómica. Esto es posible ya que alterna zonas comunes a todos los organismos, de las cuales se conoce su secuencia (conservadas), con regiones que presentan variaciones o cambios en el tiempo (variables). Las regiones conservadas son útiles para diseñar cebadores universales que permitan la amplificación de las diversas zonas hipervariables. Por otro lado, las nueve regiones (V1-V9) menos conservadas o hipervariables son las que aportan la mayor información útil para estudios de filogenia y taxonomía [1][2]. La variabilidad de estas zonas es suficiente para diferenciar no solo los organismos más alejados, sino también los más próximos.

Entendemos por secuenciación del ADN al proceso que nos permite determinar la secuencia de nucleótidos de una muestra de ADN. A finales de la década de los 70 se empezaron a obtener las primeras secuencias utilizando las denominadas tecnologías de secuenciación de primera generación. Uno de los métodos más conocidos de secuenciación de primera generación es el de Sanger [4]. Dicha técnica consta de una amplificación del gen que queremos estudiar mediante la reacción en cadena de la polimerasa (PCR por las siglas en inglés de "*polymerase chain reaction*").

En los años siguientes, se llevó a cabo el desarrollo de las *tecnologías de secuenciación de segunda o de próxima generación* [5] (NGS por sus iniciales en inglés *Next Generation Sequencing*). Estas se caracterizaron por permitir una paralelización masiva y presentar mejoras en la automatización, en la velocidad y en los costes. Las dos tecnologías de NGS más comúnmente utilizadas han sido la pirosecuenciación 454 (actualmente, en desuso) [6] e Illumina [7].

Gracias a estas técnicas de secuenciación masiva se generalizó el uso del gen 16S como gen diagnóstico para la identificación taxonómica, al ser éste un marcador filogenético para la identificación bacteriana.

En la literatura están descritos muchos pares de cebadores que funcionan adecuadamente para las diferentes zonas conservadas del gen 16S rRNA [8]. Sin embargo, estos cebadores suelen ser cebadores genéricos que no permiten seleccionar familias o géneros bacterianos de interés [9].

En muchos ámbitos de la investigación sobre la diversidad bacteriana es necesario tener la capacidad de identificar grupos concretos de bacterias, así como rechazar otros grupos para diferentes niveles taxonómicos. Determinar adecuadamente las especies bacterianas presentes en una muestra cualquiera permite, por ejemplo, asociar en un humano, animal o planta, una patología concreta a partir de la presencia o no de una especie bacteriana concreta. En estos casos, las tecnologías de secuenciación pueden condicionar al tipo de cebadores que se pueden utilizar, por ejemplo, los cebadores serán diferentes en función de la longitud de secuenciación admitida en el secuenciador seleccionado. Si se emplea una plataforma de secuenciación que genera

secuencias de un tamaño inferior a la longitud de los fragmentos que puede amplificar un determinado par de cebadores, éstos no serán válidos para esa plataforma.

En este trabajo se propone una herramienta, llamada *16SPyPrimer*, que permita realizar fácilmente un análisis de la cobertura de cebadores para conjuntos específicos de bacterias. Este software permite a los investigadores modificar cebadores ya conocidos para encontrar otros más específicos según sus necesidades.

Existe una herramienta similar a la que se propone: *testprime [8]*, que se puede encontrar en la página web de *Arb-Silva*. Este software realiza el análisis de cobertura sobre una base de datos preespecificada que incluye más de 150 mil cepas bacterianas, lo cual no permite un análisis específico. Además, *testprime* no devuelve datos de interés como puede ser el número de pares de bases existente entre las secuencias captadas por el par de cebadores, que resulta importante para saber si se puede o no utilizar una tecnología de secuenciación concreta.

Para obtener resultados de investigación diferenciados, es muy importante una adecuada selección de los cebadores y conocer su cobertura, es decir, conocer para cada par de cebadores analizados las especies bacterianas que se podrán identificar y las que no se podrán identificar. Por ello, resulta interesante que exista un software que permita a los investigadores tener una herramienta *in silico*, que sea fácilmente configurable para el usuario y que permita realizar de manera sencilla un análisis de la cobertura de cebadores para conjuntos específicos de bacterias.

Las herramientas disponibles hasta ahora, como la ya mencionada *testprime*, utilizan datos de genes 16S rRNA. Sin embargo, una novedad importante que aportará el software desarrollado en este TFM es incluir la posibilidad de evaluar genomas bacterianos completos. Además, esta herramienta se desarrollará en Python [10], que es un lenguaje sencillo, actual, ampliamente usado y que permite futuras modificaciones o ampliaciones.

1.2 Objetivos del trabajo

1.2.1 Objetivos generales

1. Analizar la cobertura de un conjunto de cebadores sobre un conjunto de genomas o genes 16S propuestos por el usuario a través de un software desarrollado en Python.

2. Permitir la configuración de la herramienta por parte del usuario, asegurando su usabilidad.

3. Crear un paquete de Python con la funcionalidad del software que permita su sencilla instalación en diversos entornos y sistemas operativos.

1.2.2 Objetivos específicos

1. Analizar la cobertura de un conjunto de cebadores sobre un conjunto de genomas o genes 16S propuestos por el usuario a través de un software desarrollado en Python.

    a. Devolver la cobertura total de un conjunto de pares de cebadores establecido por el usuario.

    b. Proporcionar la longitud del fragmento entre ambos cebadores (directo e inverso) dentro del gen 16S rRNA.

    c. Evaluar la herramienta mediante conjuntos de genes 16S y genomas completos bacterianos descargados del GenBank [11].

2. Permitir la configuración y usabilidad del software por parte del usuario.

    a. Admitir los ficheros y rutas como parámetros de entrada, así como su configuración mediante argumentos por línea de comandos.

    b. Establecer diversos parámetros por parte del usuario (por ejemplo, *mismatches* máximos).

    c. Devolver los resultados en diferentes carpetas y/o ficheros, e incluso por pantalla.

3. Crear un paquete de Python con la funcionalidad del software que permita su sencilla instalación en diversos entornos y sistemas operativos.

    a. Permitir la fácil integración con otros módulos de Python (compatibilidad).

    b. Utilizar otros módulos Python que permitan lograr los objetivos planteados.

    c. Disponer de un control de versiones de los módulos utilizados para garantizar la integración [12].

    d. Cumplir los requisitos para poder subirse a repositorios oficiales de paquetes de Python (por ejemplo, PyPi [13]).

1.3 Enfoque y método seguido

Uno de los aspectos más importantes para la realización del TFM es la formación en los conceptos más biológicos de la temática, además de la búsqueda y formación en paquetes de Python adecuados.

Por ello, las primeras etapas del proyecto fueron dedicadas a la formación en diversos ámbitos, así como al análisis de las necesidades del software. Se buscaron paquetes de Python ya existentes susceptibles de ser utilizados en el desarrollo del software, agilizando la creación del programa.

Una vez se especificaron las entradas y salidas del programa, así como el flujo que se debe realizar sobre los datos, se comenzó con la primera aproximación del software implementando las funcionalidades más básicas.

Para las pruebas iniciales se empleó un conjunto pequeño de datos para comprobar el correcto funcionamiento de la herramienta de forma controlada. Una vez la versión inicial se encuentre estable, se irán añadiendo funcionalidades según su prioridad, y repitiendo el proceso de pruebas.

1.4 Planificación del Trabajo

La gestión de la planificación temporal se realizó con la herramienta Microsoft Project [14].

1.4.1 Tareas

Este proyecto de fin de máster se desarrolló en tres fases: desarrollo de la funcionalidad (primera fase); usabilidad, configurabilidad y pruebas complejas (segunda fase); y memoria, presentación y defensa (tercera fase).

La primera fase engloba las tareas relacionadas con el análisis y el desarrollo de las funcionalidades vitales para el correcto funcionamiento del programa. La segunda fase incluye el desarrollo de las partes menos prioritarias y la realización de pruebas generales sobre el código. La tercera fase contiene la realización de la documentación final: memoria, presentación y vídeo-presentación para la defensa.

Se pueden agrupar las tareas concretas en seis grandes bloques que se encontrarán en las distintas fases: análisis (1), desarrollo (3), pruebas (1) y documentación y defensa (1).

Las tareas a continuación listadas en las diversas fases contienen los cambios y alteraciones llevados a cabo durante el desarrollo del proyecto.

1.4.1.1 Primera fase: desarrollo de la funcionalidad

Análisis

- Análisis en el ámbito de la biología: estudiar los conceptos necesarios para la realización del TFM (cebador, nomenclatura IUPAC para las bases nucleotídicas [15], gen 16S, y demás términos detectados en el análisis).

- Análisis de la funcionalidad: estudiar el alcance de la herramienta y la funcionalidad deseada.

- Análisis en el ámbito de la informática: paquetes útiles para el desarrollo eficiente de la aplicación.

Desarrollo de la funcionalidad básica

- Generación de archivos de prueba sencillos para pruebas: múltiples genes 16S y un par de cebadores.

- Análisis de cobertura sencillo utilizando genes 16S.

- Generación de archivo de estadísticas sencillo: cobertura total y distancia entre cebador directo e inverso.

- Pruebas del desarrollo realizado.

Ampliación de las funcionalidades

- Generación de archivos de prueba complejos: múltiples genomas (GenBank) y múltiples cebadores.

- Análisis de cobertura con genomas completos.

- Generación de archivos de estadísticas más completos: cobertura total y distancia entre los cebadores directo e inverso de los genomas.

- Recogida de la ruta donde se encuentren los archivos a analizar.

- Inclusión de un *logger* para la obtención de información sobre la ejecución.

- Pruebas del desarrollo realizado.

1.4.1.2 Segunda fase: usabilidad, configurabilidad y pruebas complejas

Mejoras de usabilidad

- Configuración por parte del usuario a través de línea de comandos:
    - Recogida del número de *mismatches* máximos permitidos en el análisis.
    - Recogida del tipo de datos analizados: genes 16S o genomas.
    - Recogida de parámetros opcionales.
- Pruebas del desarrollo realizado.

Mejoras de funcionalidad

- Uso de los cebadores para buscar en la hebra negativa.
- Devolución de todos los genes amplificados.
- Desglose de los archivos y nueva información.
- Dos tipos de fuentes de genomas: CSV con identificadores GenBank o carpeta con FASTA.
- Depuración de los parámetros de entrada.
- Pruebas del desarrollo realizado.

Pruebas y reflexión

- Pruebas generales sobre el código: uso de una amplia batería de datos.
- Análisis de los resultados obtenidos:
    - Detección y solución de los fallos graves que pudieran impedir el funcionamiento correcto.
    - Detección de fallos leves y posibles mejoras.

1.4.1.3 Tercera fase: memoria, presentación y defensa

Memoria, presentación y defensa del TFM

- Redacción de la memoria.
- Creación de la presentación.
- Preparación y grabación de la defensa del TFM.

1.4.2 Calendario

Se incluyen en esta sección el listado de tareas imputadas en MS Project para la planificación junto con su duración y fecha estimadas, así como los hitos al final de las fases principales, y el diagrama de Gantt derivado de dicha planificación.

Tabla 1: Lista de tareas

Nombre de tarea	Duración	Comienzo	Fin
Primera fase	**26 días**	**mar 17/03/20**	**mar 21/04/20**
Análisis	**5 días**	**mar 17/03/20**	**lun 23/03/20**
Análisis en el ámbito de la biología	1 día	mar 17/03/20	mar 17/03/20
Análisis de la funcionalidad	2 días	mié 18/03/20	jue 19/03/20
Análisis en el ámbito de la informática	2 días	vie 20/03/20	lun 23/03/20
Fin de la fase 1.1	0 días	lun 23/03/20	lun 23/03/20
Desarrollo de la funcionalidad básica	**10 días**	**mar 24/03/20**	**lun 06/04/20**
Generación de archivos de prueba sencillos	1 día	mar 24/03/20	mar 24/03/20
Análisis de cobertura sencillo utilizando genes 16S	5 días	mié 25/03/20	mar 31/03/20
Generación de archivo de estadísticas sencillo	2 días	mié 01/04/20	jue 02/04/20
Pruebas del desarrollo realizado	2 días	vie 03/04/20	lun 06/04/20
Fin de la fase 1.2	0 días	lun 06/04/20	lun 06/04/20
Ampliación de las funcionalidades	**11 días**	**mar 07/04/20**	**mar 21/04/20**
Generación de archivos de prueba complejos	1 día	mar 07/04/20	mar 07/04/20
Análisis de cobertura global con genomas completos	3 días	mié 08/04/20	vie 10/04/20
Generación de archivos de estadísticas más completos	3 días	lun 13/04/20	mié 15/04/20
Recogida de la ruta donde se encuentren los archivos a analizar	1 día	jue 16/04/20	jue 16/04/20

Inclusión de un *logger* para información sobre la ejecución	1 día	vie 17/04/20	vie 17/04/20
Pruebas del desarrollo realizado	2 días	lun 20/04/20	mar 21/04/20
Fin de la fase 1.3	0 días	mar 21/04/20	mar 21/04/20
Fin de la primera fase	0 días	mar 21/04/20	mar 21/04/20
Segunda fase	**26 días**	**mié 22/04/20**	**mié 27/05/20**
Mejoras de usabilidad	**26 días**	**mié 22/04/20**	**mié 27/05/20**
Configuración por parte del usuario a través de línea de comandos	**3 días**	**mié 22/04/20**	**vie 24/04/20**
Recogida del número de *mismatches* máximos permitidos en el análisis	1 día	mié 22/04/20	mié 22/04/20
Recogida del tipo de datos analizados	1 día	jue 23/04/20	jue 23/04/20
Recogida de parámetros opcionales	1 día	vie 24/04/20	vie 24/04/20
Pruebas del desarrollo realizado	**1 día**	**lun 27/04/20**	**lun 27/04/20**
Fin de la fase 2.1	0 días	lun 27/04/20	mié 27/05/20
Mejoras de funcionalidad	**5 días**	**lun 27/04/20**	**vie 01/05/20**
Uso de cebadores para buscar en la hebra negativa	0,5 días	lun 27/04/20	lun 27/04/20
Devolución de todos los fragmentos amplificados	0,5 días	lun 27/04/20	lun 27/04/20
Desglose de los archivos y nueva información	2,5 días	mar 28/04/20	jue 30/04/20
Dos tipos de fuentes de genomas: CSV con GenBank ids o carpeta con FASTA	1 día	jue 30/04/20	vie 01/05/20
Depuración de los parámetros de entrada	0,5 días	vie 01/05/20	vie 01/05/20
Pruebas del desarrollo realizado	**3 días**	**lun 04/05/20**	**mié 06/05/20**
Fin de la fase 2.2	0 días	mié 06/05/20	mié 06/05/20
Pruebas y reflexión	**5 días**	**jue 07/05/20**	**mié 13/05/20**
Pruebas generales sobre el código	3 días	jue 07/05/20	lun 11/05/20
Análisis de los resultados obtenidos	**2 días**	**mar 12/05/20**	**mié 13/05/20**
Detección y solución de los fallos graves que impidan el funcionamiento correcto	1 día	mar 12/05/20	mar 12/05/20
Detección de fallos leves y posibles mejoras	1 día	mié 13/05/20	mié 13/05/20
Fin de la fase 2.3	0 días	mié 13/05/20	mié 13/05/20
Fin de la segunda fase	0 días	mié 13/05/20	mié 13/05/20
Tercera fase	**30 días**	**jue 14/05/20**	**mié 24/06/20**
Memoria, presentación y defensa del TFM	**30 días**	**jue 14/05/20**	**mié 24/06/20**

Redacción de la memoria	25 días	jue 14/05/20	mié 17/06/20
Fin de la fase 3.1	0 días	mié 17/06/20	mié 17/06/20
Creación de la presentación	2 días	jue 18/06/20	vie 19/06/20
Fin de la fase 3.2	0 días	vie 19/06/20	vie 19/06/20
Preparación y grabación de la defensa del TFM	3 días	lun 22/06/20	mié 24/06/20
Fin de la fase 3.3	0 días	mié 24/06/20	mié 24/06/20

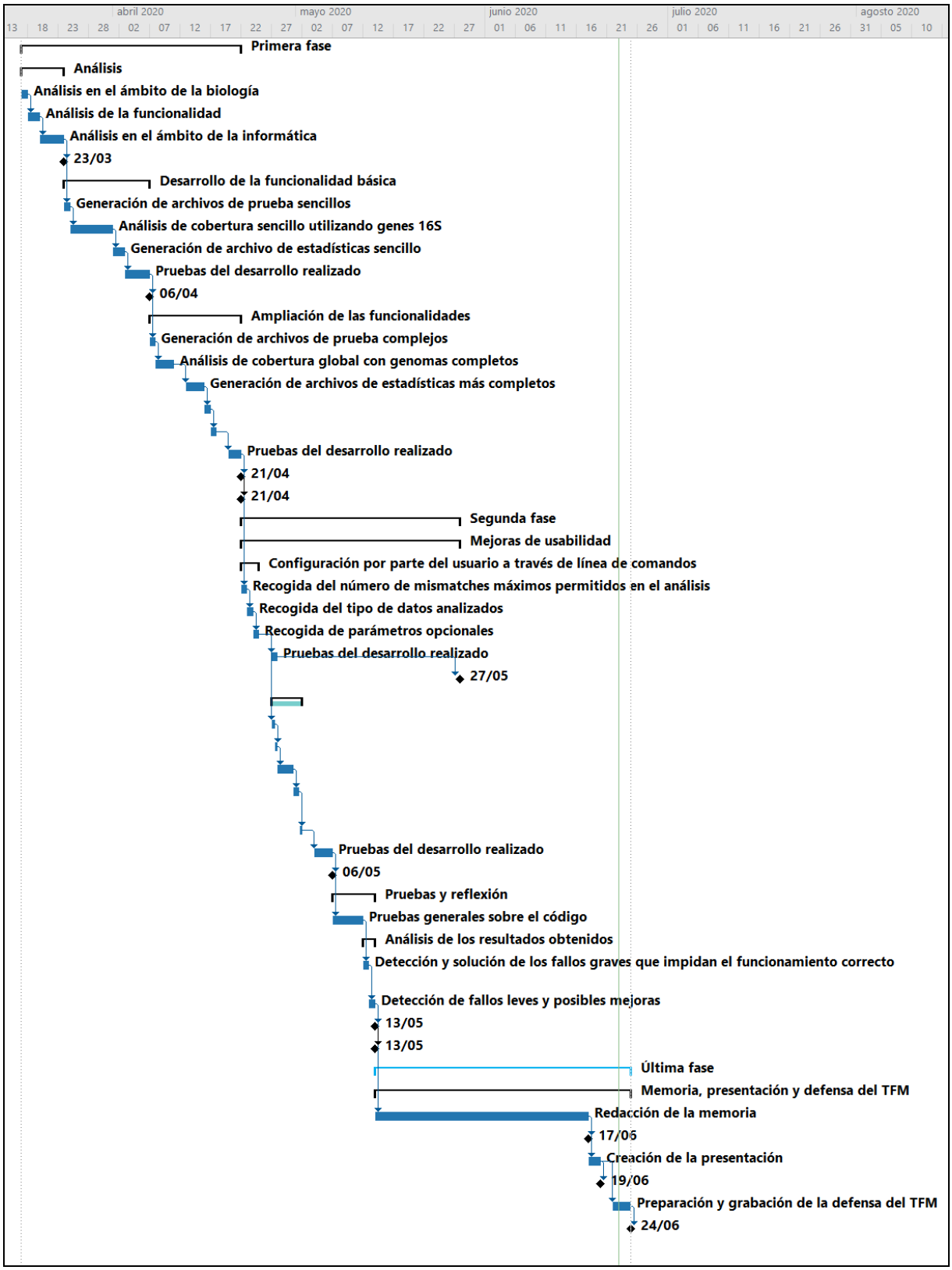

Ilustración 2: Diagrama de Gantt

1.4.3 Hitos

Al final de cada una de las fases descritas anteriormente hay un hito. Estos pueden corresponderse con la entrega de una PEC, o simplemente con la finalización de una fase importante en el proyecto y sin la cual no se podría avanzar.

Inicialmente se disponía de unas fechas diferentes que se vieron modificadas durante el desarrollo del proyecto. Para la segunda fase se dispuso de 2 semanas extra, ampliando la fecha de entrega de la PEC 3 al 01/06. Sin embargo, como el trabajo se estaba desarrollando correctamente según la planificación inicial, se mantuvo la fecha para la segunda fase y se dedicó el tiempo extra a la tercera fase.

Tabla 2: Hitos del proyecto

FASE	HITO	FECHA
PRIMERA FASE	Hito de fin de fase 1.1: análisis	23/03/2020
	Hito de fin de fase 1.2: desarrollo de la funcionalidad básica	06/04/2020
	Hito de fin de fase 1.3: ampliación de las funcionalidades	20/04/2020
	Hito de fin de primera fase: PEC 2	20/04/2020
SEGUNDA FASE	Hito de fin de fase 2.1: mejoras de usabilidad	27/04/2020
	Hito de fin de fase 2.2: mejoras de funcionalidad	06/05/2020
	Hito de fin de fase 2.3: pruebas y reflexión	13/05/2020
	Hito de fin de segunda fase: PEC 3	13/05/2020
TERCERA FASE	Hito de fin de 3.1: PEC 4 - redacción de la memoria	17/06/2020
	Hito de fin de 3.2: PEC 5a - creación de la presentación	19/06/2020
	Hito de fin de 3.3: PEC 5b - defensa pública del TFM	24/06/2020

En todas las fases se hace referencia a la entrega de las PEC, pero esta fecha no siempre se corresponde con la fecha límite real de las PEC. Esto se debe a que se planificó de manera que las entregas estén listas antes de la fecha real, para estar preparados ante posibles retrasos causados por algún imprevisto. Las fechas límites reales de las PEC y por tanto la fechas límites de esas fases se muestran en la siguiente tabla.

Tabla 3: Entregas del proyecto

ENTREGA	DESCRIPCIÓN	FECHA
PEC 2	Desarrollo del trabajo – Fase 1	22/04/2020
PEC 3	Desarrollo del trabajo – Fase 2	01/06/2020
PEC 4	Cierre de la memoria	24/06/2020
PEC 5A	Elaboración de la presentación	28/06/2020
PEC 5B	Defensa pública	01/07/2020 - 08/07/2020

1.4.4 Análisis de riesgos

Un riesgo es un problema potencial, es decir, puede ocurrir o no. Para la buena gestión de un proyecto se deben analizar los riesgos más graves que pueden afectar el desarrollo de este y proponer planes de contingencia.

Como en todo proyecto donde se desarrolle un software, el más preocupante es la aparición de un error o impedimento durante la creación del software que impida avanzar, y por tanto haya que replantear todo hasta el momento. Para evitar este problema lo más recomendable es asegurar un buen análisis de las funcionalidades a implementar, asegurándose periódicamente que se continúa por el buen camino, así como procurar emplear las mejores prácticas de programación posibles.

No solo eso, sino que en este TFM en especial se quiere asegurar la integración del software en diversos entornos y sistemas operativos, manteniendo un control de las versiones de todos los paquetes externos que en él se utilizan. Esto es un riesgo añadido al anterior, ya que puede darse el caso de necesitar utilizar un paquete ya existente que entre en conflicto con

otras versiones de otros paquetes que se utilizan, o que directamente no sirva para algún sistema operativo.

Otro riesgo más típico de la gestión de proyectos en general es la mala planificación temporal, subestimando el tiempo que necesita una tarea para llevarla a cabo. Para evitar este problema es necesario planificar con cautela, y dejando margen entre la finalización estimada de las tareas con la finalización del plazo de entrega. En este TFM se ha planificado el calendario de cada tarea de modo que quede un tiempo entre ellas, que podrá ser utilizado en caso de necesidad. Además, se realizaron comprobaciones periódicas del progreso del proyecto para poder replanificar en caso de haber alguna desviación en la estimación temporal inicial.

1.5 Productos obtenidos

Al finalizar este proyecto se obtienen los siguientes resultados:

- Herramienta en Python para el análisis de cobertura de cebadores.
- Memoria del proyecto.
- Presentación del proyecto.

1.6 Estructura de la memoria

Esta memoria muestra el trabajo realizado dividido en 4 capítulos troncales, además de la introducción, conclusiones, glosario y bibliografía. Estas secciones son: análisis, diseño, implementación y pruebas y reflexión, que se explican brevemente a continuación junto con sus subsecciones.

2. Análisis

2.1. Análisis en el ámbito de la biología: descripción detallada de la formación realizada en el ámbito de la biología. Incluye la familiarización con conceptos específicos de este ámbito.

2.2. Análisis en el ámbito de la funcionalidad: descripción detallada del análisis realizado sobre la funcionalidad esperada del proyecto. Incluye determinados requisitos detectados durante la realización del trabajo.

2.3. Análisis en el ámbito de la informática: descripción detallada del análisis realizado sobre diferentes paquetes de Python que puedan resultar útiles en el desarrollo de la herramienta.

3. **Diseño**
   - 3.1. **Estructura de clases:** descripción de las clases de Python creadas en la implementación del código.
   - 3.2. **Método de búsqueda de cebadores:** descripción del método de búsqueda de cebadores empleado en el análisis, tanto de genes 16S como de genomas.

4. **Implementación**
   - 4.1. **Primera fase - desarrollo de la funcionalidad:** descripción de las tareas realizadas en orden cronológico en la primera fase.
     - 4.1.1. **Desarrollo de la funcionalidad básica:** descripción del proceso de desarrollo de la funcionalidad reducida implementada de forma inicial para obtener un producto mínimo. Principalmente análisis de cebadores sobre genes 16S.
     - 4.1.2. **Ampliación de funcionalidades:** descripción del proceso de desarrollo de la funcionalidad ampliada en la que se obtiene un producto más completo. Ampliación del análisis de cebadores sobre genes 16S para que analice genomas.
   - 4.2. **Segunda fase - usabilidad, configurabilidad y pruebas complejas:** descripción de las tareas realizadas en orden cronológico en la segunda fase.
     - 4.2.1. **Mejoras de usabilidad:** descripción del proceso de inclusión de nuevas mejoras de usabilidad para la recogida de parámetros de entrada. Implican la configurabilidad de la herramienta.
     - 4.2.2. **Mejoras de funcionalidad:** descripción del proceso de inclusión de mejoras en la funcionalidad detectadas durante el desarrollo del proyecto.

5. **Pruebas y reflexión**
   - 5.1. **Pruebas del software:** descripción de las pruebas generales de verificación de la eficacia del código realizadas.

5.2. **Análisis de los resultados obtenidos:** descripción del análisis y pasos llevados a cabo a partir de los resultados obtenidos de las pruebas del software.

También se incluyen dos anexos con información útil para el usuario final sobre software elaborado.

9. Anexos

9.1. **Manual de usuario:** descripción de la instalación y ejecución del software. Listado de los comandos y opciones de ejecución disponibles.

9.2. **Estructura de archivos de resultados:** descripción de la nomenclatura y estructura de los archivos generados según el tipo de análisis de cebadores (sobre genes 16S o genomas).

2. Análisis

Previo al proceso de implementación es necesario dedicar una cantidad de tiempo suficiente al análisis de las necesidades del proyecto y recopilar exactamente qué se realizará durante el desarrollo.

En este trabajo se combina la informática y la genómica, por lo que también es importante controlar los conocimientos pertenecientes a ambos ámbitos que sean precisos para una realización correcta y eficiente del proyecto.

2.1 Análisis en el ámbito de la biología

Resulta necesario para la realización del proyecto estudiar los conceptos biológicos que se aplicarán en desarrollo, destacando: cebador, gen 16S y nomenclatura IUPAC.

Se comienza el estudio con los cebadores ya que su uso y aplicación puede llegar a ser complejo si no se comprenden correctamente. Un cebador (en inglés, *primer*) es una secuencia corta de ADN de cadena simple que se utiliza en una reacción en cadena de la polimerasa (PCR) [16]. En el método PCR se emplea un par de cebadores para hibridar con el ADN de la muestra y definir la región del ADN que será amplificada. Este par de cebadores está compuesto por el cebador directo y el inverso.

Ambos cebadores se deberán recibir en sentido 5’ 3’ para poder aplicarles unas transformaciones necesarias según se esté analizando la hebra positiva o negativa del genoma. Para la hebra positiva, que es la que se recibe directamente del usuario, el cebador directo se empleará tal y como se recibe, sin realizarle ninguna transformación, pero el cebador inverso se empleará haciendo su inversa complementaria. Para la hebra negativa se invertirá la posición de los cebadores a la hora de analizar la secuencia y se hará la inversa complementaria del cebador originalmente directo.

Modificar los cebadores para analizar ambas hebras en vez de realizar la inversa complementaria del genoma supone una mayor eficiencia y menor carga computacional, aumentando la velocidad a la hora de ejecutar la herramienta.

En el caso de analizar la cobertura sobre los genes no será necesario analizar ambas hebras, sino directamente la secuencia del gen proporcionada por el usuario.

Es necesario también analizar la nomenclatura IUPAC para las bases nucleotídicas, puesto que cuando se haga el análisis de secuencias con los cebadores se deberán admitir todas aquellas secuencias que cumplan cualquiera de las posibilidades generadas por dicho cebador según esta notación. Es decir, si un cebador contiene alguna de las letras del código

IUPAC, significa que en esa posición tendrá uno de los nucleótidos asociados a esa letra según la nomenclatura. Se puede ver la tabla 4 con las correspondencias.

Tabla 4: Código IUPAC para las bases nucleotídicas. Fuente: [15]

Symbol	Bases
G	G
A	A
T	T
C	C
R	G or A
Y	T or C
M	A or C
K	G or T
S	G or C
W	A or T
H	A or C or T
B	G or T or C
V	G or C or A
D	G or A or T
N	G or A or T or C

El gen 16S, como ya se comentó en la introducción, resulta de gran interés al ser un marcador filogenético para la identificación bacteriana. Este gen posee zonas variables delimitadas por zonas conservadas, de las cuales se conoce la secuencia. En las regiones más variables es donde se encuentra la información más útil para la diferenciación de organismos. Sin embargo, las regiones conservadas, al ser conocidas, posibilitan diseñar cebadores universales que permiten amplificar las regiones más variables del gen.

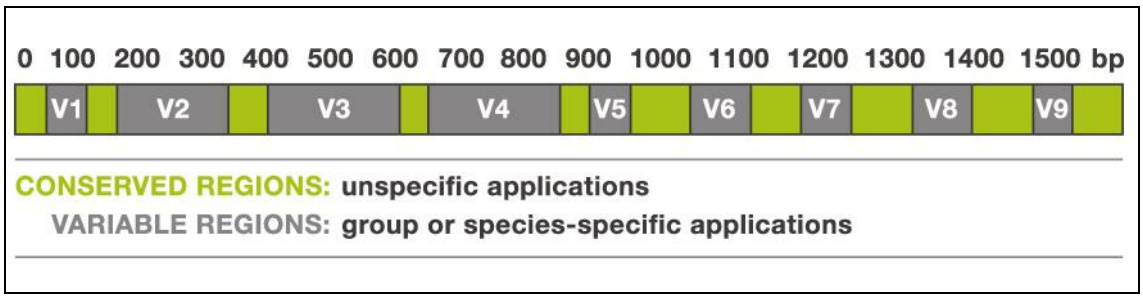

Ilustración 3: Regiones del gen 16S. Fuente: [17]

Es necesario tener en cuenta que con diferentes cebadores se pueden amplificar regiones distintas del ADN. Según qué región se amplifique, los fragmentos encontrados pueden ser más pequeños (por ejemplo, 200pb[1]) o más grandes (por ejemplo, 500pb). Además, en un genoma pueden encontrarse diversas copias, por lo que los cebadores podrían amplificar más de un fragmento. Estos datos resultan importantes a la hora de validar los resultados obtenidos mediante las pruebas realizadas sobre el software.

2.2 Análisis de la funcionalidad

Este análisis se cubre en la primera fase del proyecto, que engloba el análisis y el desarrollo de la funcionalidad básica, aunque algunas de las necesidades del proyecto se van detectando a lo largo del desarrollo, y por tanto se va adaptando la planificación y listado de tareas inicial.

Se analizan las necesidades del proyecto en relación con el software que se va a implementar, determinando principalmente qué entradas necesita y qué salidas devolverá y en qué condiciones.

La herramienta analizará la cobertura de al menos un par de cebadores dados como entrada sobre un grupo de genes 16S o genomas, según indique el usuario.

Se buscarán los fragmentos amplificados, que se corresponderían con posibles genes 16S, por cada par de cebadores en ambos casos. Cuando se analice la cobertura sobre los genes solo se podrán encontrar como máximo un gen, el propio que se está analizando, sin embargo, analizando sobre genomas se podrán encontrar múltiples genes.

Se determinará que un gen o genoma está cubierto con un cebador cuando se encuentra al menos un gen en él. Este concepto de cobertura no se debe confundir con su acepción más común: cuántas veces ha sido secuenciado un nucleótido concreto de una secuencia.

A partir de este análisis se determinan una serie de hechos concretos sobre cómo deben ser las entradas:

- El formato del archivo de cebadores será CSV y no tendrá cabecera. Cada par de cebadores estará en una línea diferente.

- El formato del archivo de genes será FASTA.

- El archivo de genomas se podrá dar de dos formas: un listado CSV con los identificadores GenBank de los genomas o una carpeta con los archivos FASTA que contienen los genomas.

- Los archivos FASTA deben tener cabecera + secuencia.

[1] Abreviación de pares de bases.

- Solo se aceptarán los cebadores en notación IUPAC, pero sin *gaps*.

- Se asumirá que los cebadores se adjuntan en sentido 5’ 3’.

- Para aceptar un fragmento amplificado por un par de cebadores como válido éste no deberá superar el tamaño máximo (distancia) establecido por el usuario.

- Sólo se aceptarán diferencias entre el cebador y la secuencia diana (*mismatches*) del tipo sustitución de nucleótido, no las inserciones o deleciones [18].

La mayoría de estos requisitos deberán ser cumplidos por el usuario de la herramienta, pero establecerlos permite hacer un buen control y comprobación de las entradas, así como poder implementar un programa robusto que reaccione correctamente ante datos de entrada no contemplados. Estas pautas se pondrán a disposición del usuario como parte de la documentación de la herramienta para facilitar su uso.

Además de estas entradas el usuario podrá indicar una serie de opciones para modificar las condiciones del análisis.

Dependiendo del tipo de análisis de cobertura, que puede ser sobre genes o sobre genomas, se devolverán unos resultados diferentes.

Para el análisis sobre genes se generarán como mínimo 4 archivos:

- Un archivo donde se indicará la información sobre los cebadores y genes analizados y su cobertura en relación con ellos.

- Un archivo donde se indicarán datos más específicos sobre el análisis, como la longitud del fragmento amplificado o el propio fragmento.

- Dos archivos FASTA por cada par de cebadores donde se guardarán las secuencias completas de los genes amplificados por cada par de cebadores y de los no amplificados, respectivamente.

Para el análisis sobre genomas se generarán como mínimo 2 archivos y tantas carpetas como cebadores:

- Un archivo donde se indicará la información sobre los cebadores y genomas analizados y su cobertura en relación con ellos.

- Un archivo donde se indicarán datos más específicos sobre el análisis, como la longitud de los fragmentos amplificados dentro del genoma o los propios fragmentos.

- Una carpeta por cada par de cebadores donde se guardarán en diferentes archivos FASTA los diferentes fragmentos amplificados de

cada genoma analizado. Asimismo, en cada carpeta se guardará un archivo CSV que guardará la relación de los genomas amplificados y no amplificados por cada par de cebadores.

El objetivo principal de los resultados generados es aportar la máxima información posible al usuario para que pueda realizar análisis propios a posteriori sobre los datos.

En el Anexo 9.2 se hace una descripción más exhaustiva de los datos que se guardan en cada archivo así como la nomenclatura de estos.

2.3 Análisis en el ámbito de la informática

El objetivo de este trabajo es desarrollar una aplicación en Python susceptible de convertirse en paquete en el futuro. Esta herramienta no tendrá interfaz, sino que se implementará de manera que sus funcionalidades se puedan usar mediante línea de comandos. En el Anexo 9.1 se desarrolla el manual de usuario y además se describen ejemplos de uso.

Antes de implementar el código se debe realizar una búsqueda extensa de los paquetes bioinformáticos de Python disponibles que sean útiles para el desarrollo eficiente de la aplicación. Se empleará la versión 3.7 de Python, que es reciente y estable.

Es importante que a la hora de seleccionar un paquete se compruebe su fiabilidad, que inicialmente se puede obtener a partir del número de descargas que posea y de la valoración de otros desarrolladores. Independientemente de este criterio, se deberá hacer pruebas de uso propias para verificar que funciona correctamente y cumple las necesidades que debe suplir.

A partir de la funcionalidad que se espera de la herramienta, se pueden prever muchos de los paquetes que se usarán:

- *BioPython* [19]: contiene métodos y estructuras de datos para la bioinformática. Por ejemplo, un diccionario para la notación IUPAC, que devuelve para cada letra sus posibles nucleótidos. También permite la descarga cómoda y fiable de secuencias de las bases de datos del NCBI [20].

- *argparse* [21]: permite recoger los datos introducidos por línea de comandos y establecer determinados parámetros como su obligatoriedad o valor por defecto entre otros.

- *os* [22]: permite realizar operaciones de lectura y escritura de archivos, que será útil para procesar las entradas del programa y generar las salidas.

- *regex* [23]: para la creación y aplicación de expresiones regulares. Se empleará en la búsqueda de cebadores en las secuencias.

- *csv* [24]: permite leer archivos de CSV de manera automática sin indicar previamente cuál será el delimitador de los valores empleado en estos. Permite flexibilizar la entrada al no obligar al usuario a usar un delimitador concreto, siendo por estándar la coma.

- *logging* [25]: permite generar archivos de texto que contienen información sobre la ejecución del programa, como pueden ser los pasos realizados o errores que ocurrieran durante la ejecución.

3. Diseño

3.1 Estructura de clases

En esta sección se explicará la estructura del código a nivel técnico.

Python dispone de un tipo de dato llamado clase [26], que permite describir un conjunto de objetos similares y proporciona métodos y datos que resumen las características comunes de dicho conjunto. Estas características comunes se denominan atributos. Es importante definir el concepto de instancia, que es una particularización de una clase, es decir, es un objeto creado a partir de esa clase, que se puede ver como un nuevo tipo de dato.

En este software se manejarán varios tipos de datos con frecuencia, como son los cebadores, los genes y los genomas. Por ello, resulta interesante analizar aquellas características que interesan para el desarrollo y agruparlas en una clase. Esto permite guardar y procesar la información de manera más sencilla.

Las características más importantes de un par de cebadores son la secuencia del cebador directo, la secuencia del cebador inverso, la inversa complementaria de ambas secuencias y la cobertura que éstos poseen respecto a los genes/genomas sobre los que se analizaron.

En el caso de los genes, interesa recoger su secuencia y la cabecera que se recogerá del archivo FASTA. En el caso de los genomas interesa recoger, además de la secuencia, la cabecera (que proviene de un CSV si se ha descargado del NCBI) o el nombre del archivo (si proviene de un archivo FASTA).

Todas estas clases guardarán otros datos menos característicos pero sí importantes a nivel de implementación como, por ejemplo, secuencias identificadoras para poder identificar correctamente cada una de las instancias de esa clase.

La primera clase a considerar es la que recoge la información de los cebadores, la segunda la que recopila la información de los genes, y la tercera la de los genomas. Estas tres clases son las más intuitivas y las primeras que se han de implementar, pero también resulta beneficioso crear otras, como una que recopile los datos importantes sobre la secuencia diana que encuentra un cebador u otra que recoja los datos relevantes sobre el fragmento que será amplificado a partir de un par de cebadores.

También se crearán clases para aspectos más técnicos, como son clases auxiliares para la lectura, escritura y procesado de archivos de entrada y salida o para el algoritmo de búsqueda de cebadores en los genes/genomas.

Estas clases se suelen representar en un diagrama de clases donde también se indican sus atributos. En la Ilustración 4 se muestra el diagrama de clases de este software.

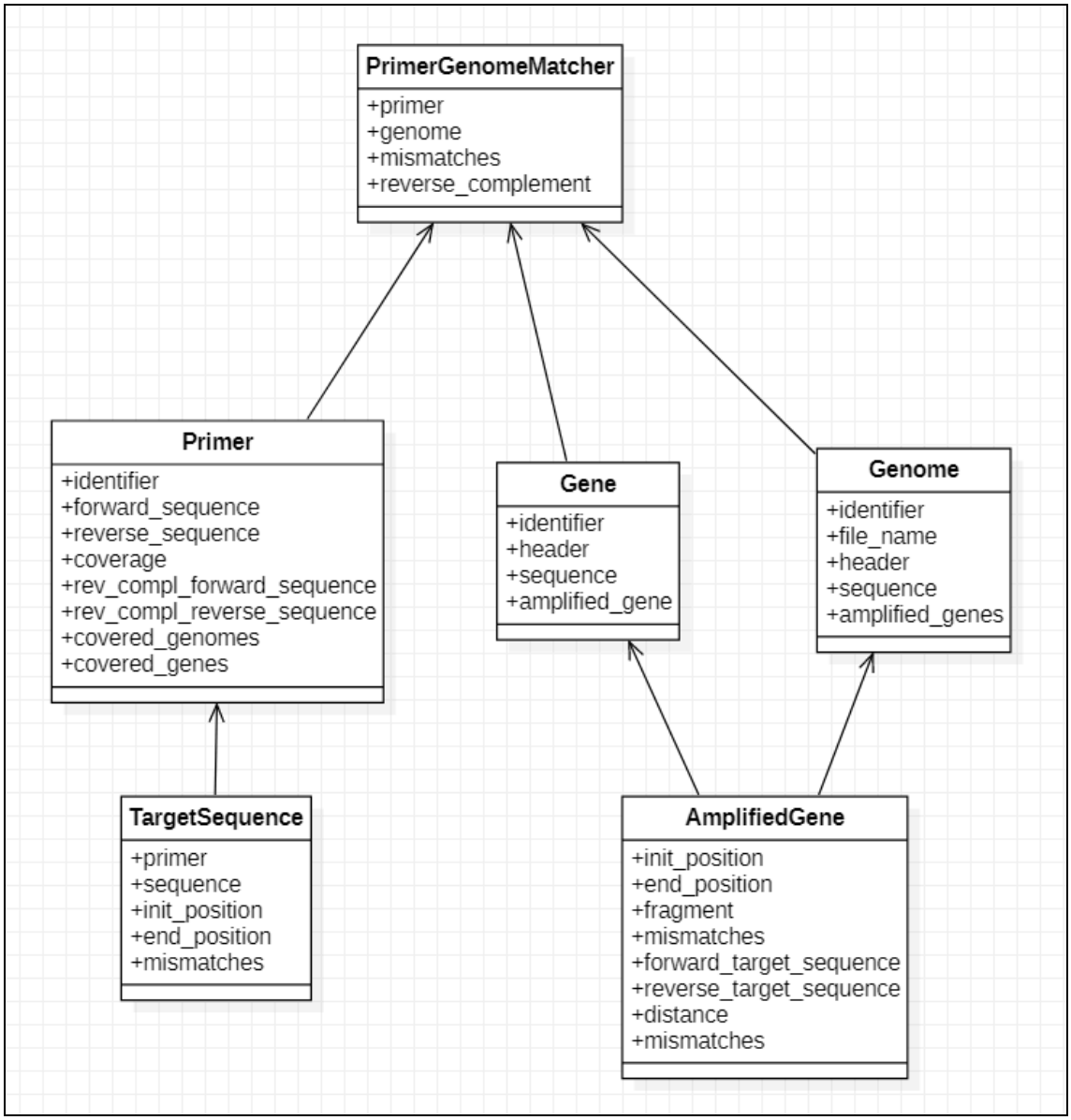

Ilustración 4: Diagrama de clases definido para el desarrollo de 16SPyPrimer

3.2 Método de búsqueda de cebadores

El análisis de cobertura de los pares de cebadores se basa en la búsqueda de éstos sobre las secuencias proporcionadas por el usuario, sean genes 16S o genomas. Por ello, es necesario idear una manera eficiente de realizar esta búsqueda.

La clave está en usar expresiones regulares, que son secuencias de caracteres que conforman patrones de búsqueda. Gracias a ellas se pueden

recorrer las secuencias en un tiempo mínimo y obtener todas las coincidencias que se encuentren con esos cebadores.

Se deberá hacer una búsqueda en cada una de las secuencias para cada cebador del par de cebadores. Una vez se obtienen todas las coincidencias, es decir, secuencias captadas por los cebadores, para el cebador directo y para el inverso, se podrán detectar los fragmentos amplificados.

Esta expresión regular deberá tener en cuenta los *mismatches* que puede establecer el usuario. Para cada cebador que se busque en la secuencia se deberá tener en cuenta dicho valor. Cuando se forme el fragmento se verificará que el conjunto de los *mismatches* de ambos no supere el máximo establecido por el usuario.

De este modo, se diseña una expresión regular como la siguiente,

$$(primer\_sequence)\{\{s<=mismatches\}\}$$

donde *primer_sequence* denotará la secuencia del cebador a analizar y *mismatches* denotará el valor máximo de diferencias entre el cebador y la secuencia diana.

Se emparejarán las secuencias captadas por los cebadores según su proximidad. La iteración empezará con las secuencias captadas por el cebador directo, y buscará todas aquellas secuencias captadas por el cebador inverso que se encuentren después del directo.

Se puede dar el caso de que se encuentren más secuencias captadas de un cebador que de otro. Se deberán descartar todas aquellas secuencias que no tengan pareja asociada.

4. Implementación

Se explicará la implementación de la herramienta siguiendo el orden temporal de realización de las tareas para poder observar la evolución real del proyecto. Se optó por realizar un desarrollo gradual de las tareas según su dificultad, empezando por los conceptos más simples para obtener un producto base sobre el que ir avanzando en etapas posteriores.

4.1 Primera fase: desarrollo de la funcionalidad

La primera fase consistió, en primer lugar, en el desarrollo de la funcionalidad básica realizando primero el análisis de cobertura de los pares de cebadores sobre los genes 16S indicados por el usuario, para posteriormente aumentar la funcionalidad hasta analizar también la cobertura sobre genomas. Durante todas las etapas de implementación se realizan pruebas para comprobar el correcto funcionamiento, por lo que también es necesario generar una serie de archivos de entrada sencillos que servirá para realizar estas pruebas.

4.1.1 Desarrollo de la funcionalidad básica

Se explica en detalle a continuación cómo se realizó cada tarea dentro de la etapa de desarrollo de la funcionalidad básica.

Generación de archivos con datos sencillos para pruebas

Múltiples genes 16S y un par de cebadores

A la hora de comenzar a desarrollar la aplicación es importante realizar pruebas con datos pequeños y controlados. Por ello, se genera un archivo CSV para los cebadores que solo dispone de un par de cebadores, así como un archivo FASTA para los genes 16S que solo dispone de dos genes 16S.

Los genes 16S se obtuvieron del HOMD (*Human Oral Microbiome Database*) [27], que posee archivos FASTA con múltiples genes 16S pertenecientes a bacterias de la microbiota oral.

Análisis de cobertura sencillo utilizando genes 16S

Para el análisis de cobertura hay dos piezas clave que facilitan la búsqueda: el diccionario de datos para la notación IUPAC del paquete BioPython y las expresiones regulares.

Basándose en esto, el proceso a seguir para preparar los cebadores es el siguiente:

- Preparar la expresión regular:

    - Se forma una expresión regular a partir del cebador reemplazando los nucleótidos con indeterminaciones según la notación IUPAC por sus posibles sustituciones. Por ejemplo: `ACGTCNGT` por `ACGTC[ACGT]GT`. Cuando se busque este patrón sobre una secuencia, se podrán detectar todas las secuencias que comiencen por `ACGTC`, que continúen por A, C, G o T, y que terminen en `GT`.

    - Se añaden unos atributos adicionales a la expresión regular para que permita el número de *mismatches* establecido por el usuario, incluyendo `{s<=MISTMATHES}` al final de la expresión regular.

- Realizar la búsqueda de cebadores sobre las secuencias:

    - Primero se busca con el cebador directo y luego con el inverso. En ambos casos se devuelven los datos relevantes de la secuencia diana captada, si es el caso (posiciones en el genoma y secuencia exacta encontrada).

    - Se busca para cada secuencia captada por el cebador directo la secuencia captada por el cebador inverso correspondiente. Con las parejas formadas, se obtienen los posibles fragmentos amplificados por el par de cebadores. El número total de *mismatches* de ambas secuencias captadas por los dos cebadores no debe superar el número de *mismatches* establecido por el usuario. En esta situación, con genes 16S, para cada par de cebadores sólo podrá formarse una pareja, de ser el caso.

Para la búsqueda de la expresión regular en las secuencias se parte de un código existente en el módulo SeqUtils [28] perteneciente al paquete de BioPython. Este código sirve como base funcional para el software pero es necesario realizar una serie de modificaciones de éste para incluir los *mismatches* o generar los objetos con la información que interesa mantener de la búsqueda.

Generación de archivo de estadísticas sencillo

cobertura total y longitud del fragmento amplificado por el par de cebadores

Mediante el programa se genera un archivo CSV para los resultados iniciales. En él se incluye, para cada cebador, la longitud del fragmento amplificado (0 si no encuentra) y su cobertura total (número de genes 16S que amplifica entre el total de número de genes que se analizaron).

Pruebas del desarrollo realizado

Se emplean los archivos generados con anterioridad para hacer una batería de pruebas. Al estar analizando genes 16S, es más fácil comprobar si funciona correctamente que con genomas, ya que se puede analizar fácilmente con BLAST [29] u otra herramienta similar. En el gen 16S solo se podrá amplificar un fragmento con cada par de cebadores, perteneciente al propio gen que se está analizando.

4.1.2 Ampliación de las funcionalidades

La siguiente etapa dentro de esta fase consiste en la ampliación de las funcionalidades existentes para realizar un análisis más complejo, el análisis de la cobertura sobre los genomas.

Generación de archivos de prueba complejos

Múltiples genomas (GenBank) y múltiples cebadores

Se conserva el archivo creado anteriormente para un par de cebadores, y se genera a partir de él otro con más pares de cebadores (cuatro en total). Se genera también un archivo FASTA con un genoma. Según se hacen las pruebas se va aumentando gradualmente hasta llegar a 20 genomas en total.

Estos genomas se obtienen de la base de datos del NCBI, empleando la herramienta E-utilities [30][31] para realizar consultas mediante URLs. Se indicarán en la petición los identificadores GenBank de los genomas que se desean obtener separados por comas. Por ejemplo:

```
https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nucleotide&id=CP006958,HE798385&rettype=fasta
```

Este método de obtención de los genomas resulta útil para la implementación de obtención de genomas del NCBI si se indican a través de un archivo CSV con los identificadores GenBank.

Análisis de cobertura con genomas completos

Se parte de la lógica realizada para el análisis de cebadores en genes 16S y se reutiliza el proceso de creación de la expresión regular para el análisis de cebadores, pero es necesario modificar el método de búsqueda de cebadores sobre las secuencias, adaptándolo a las nuevas necesidades con los genomas.

A diferencia de con los genes 16S, al buscar tanto con el cebador directo como con el cebador inverso, se devuelve una lista con todas aquellas coincidencias, es decir, las secuencias diana que capten ambos cebadores. Los cebadores directo e inverso pueden captar un número diferente de secuencias diana, por ejemplo, en un genoma el directo podría captar 3 y el

inverso 2. Por ello, es necesario realizar un filtrado y emparejamiento de las secuencias diana recuperadas en las listas.

Para comprobar la validez de las secuencias encontrada se busca, para cada secuencia diana del cebador directo, una secuencia diana del cebador inverso válida. Aquellas que se queden sin pareja, serán descartadas.

Una vez se tienen las parejas formadas, se obtienen los posibles fragmentos amplificados por el par de cebadores, pero también es importante controlar que el número de *mismatches* total en las secuencias diana de los dos cebadores no supere el máximo establecido, ya que al buscar los cebadores por separado en la secuencia se están permitiendo el doble de fallos. Es decir, se permitirían tantos fallos como indique el usuario por cebador, por lo que hay que filtrar a posteriori.

Este análisis se realiza sobre la hebra positiva y la negativa del genoma, que se obtiene realizando la inversa complementaria sobre éste.

Generación de archivos de estadísticas más completos

Cobertura total y longitud entre los cebadores directo e inverso de los genomas

Se genera un archivo CSV inicial para las estadísticas. En él se incluirá, para cada par de cebadores, todos los fragmentos que amplifica en cada genoma, indicando la longitud de este fragmento (entre cebador directo e inverso) y la cobertura de dicho par de cebadores. La cobertura, al ser global, se devuelve para cada par de cebadores.

Además, se devuelve algún dato más sobre cada fragmento amplificado:

- Secuencia del fragmento amplificado.
- Secuencia diana de cada cebador en el fragmento, ya que el cebador puede contener nucleótidos indeterminados y resulta interesante guardar la secuencia exacta encontrada.
- Número de *mismatches* detectados en el par de cebadores, de haberlos.

Recogida de la ruta donde se encuentren los archivos a analizar

Se recogen las rutas de los archivos a analizar mediante línea de comandos, para facilitar el desarrollo y la realización de pruebas al no tener que indicar en el propio código estos directorios o archivos.

Inclusión de un *logger* para la obtención de información sobre la ejecución

Se incluye un módulo que permite crear archivos de *logs* para recoger información sobre el progreso y errores de la ejecución del programa.

El módulo empleado es *logging*, ya que permite configurar desde el principio cuál será la ruta y los archivos específicos en los que se guardarán estos datos, así como permitir dar formato a los mensajes que se mostrarán en ellos.

Pruebas del desarrollo realizado

Al igual que con el análisis de genes, se emplean los archivos generados con anterioridad para hacer una batería de pruebas. Para comprobar si funciona correctamente se analizan los cebadores con BLAST sobre un número limitado de genomas. También se emplean editores de código como VSCode [32] para realizar búsquedas literales sobre las secuencias completas, ya que estos editores permiten buscar empleando expresiones regulares.

4.2 Segunda fase: usabilidad, configurabilidad y pruebas complejas

La segunda fase consistió en las mejoras de usabilidad para permitir al usuario introducir más opciones mediante línea de comandos, proporcionando configurabilidad al software. También se aplicaron una serie de mejoras sobre el código para aumentar su eficiencia y ampliar la información devuelta al usuario.

Al final de esta fase se procedió a realizar pruebas más completas y generales sobre el código, pero éstas se detallarán en su propia sección.

4.2.1 Mejoras de usabilidad

En esta etapa se implementan una serie de funcionalidades relacionadas con la recogida de nuevos parámetros mediante línea de comandos.

Recogida del número de *mismatches* máximos permitidos en el análisis

Se permite al usuario indicar de manera opcional el número máximo de diferencias entre el cebador y la secuencia diana (*mismatches*). Con esta opción se permite al usuario realizar un análisis más flexible que le permita evaluar y encontrar mejores cebadores a partir de las diferencias existentes entre el cebador y la secuencia diana.

El valor por defecto pensado inicialmente era 1, pero en esta fase se determina que debería ser 0 y que sea decisión del usuario aumentarlo hasta un máximo de 5 *mismatches*.

Recogida del tipo de datos analizados

El usuario debe indicar obligatoriamente el tipo de secuencias sobre las que se calculará la cobertura de los cebadores. Pueden ser tanto genes 16S como genomas.

Recogida de parámetros opcionales

Para hacer más flexible y configurable la herramienta se añaden tres parámetros opcionales nuevos:

- Tamaño mínimo del fragmento amplificado por los pares de cebadores: por defecto será de 200pb.

- Tamaño máximo del fragmento amplificado por los pares de cebadores: por defecto será de 600pb.

- Ruta a la carpeta donde se deseen guardar los resultados: si no se especifica se creará una carpeta llamada 'results' en el directorio donde se ejecute el programa.

4.2.2 Mejoras de funcionalidad

En esta etapa se implementan ciertos cambios y se añaden pequeñas modificaciones para mejorar la eficiencia y funcionamiento general de la herramienta.

Uso de los cebadores para buscar en la hebra negativa

Se emplea la secuencia complementaria inversa de los cebadores en vez de la complementaria inversa del genoma para buscar sobre la hebra negativa, haciendo más eficiente el análisis de cobertura.

Es importante recordar que un requisito a la hora de ejecutar el software es introducir ambos cebadores en sentido 5' 3'. Partiendo de este hecho, se realizan transformaciones sobre los cebadores para poder buscar en ambas hebras.

Para la búsqueda sobre la hebra positiva, primero se buscan coincidencias de la secuencia del cebador directo en el sentido 5' 3', que es el original, y después coincidencias para el cebador inverso, pero tomando la inversa complementaria de éste. Sin embargo, para analizar la hebra negativa se "invierten" los cebadores, es decir, se busca primero con el cebador inverso (que pasa a actuar como directo) en el sentido proporcionado por el usuario, que debe ser 5' 3', y después con el directo (que pasa a actuar como inverso), haciendo la inversa complementaria del directo.

Devolución de todos los fragmentos amplificados

Interesa devolver todos los fragmentos amplificados por cada par de cebadores, independientemente de que estén o no en el rango de longitud establecido por el usuario (por defecto, entre 200pb y 600pb). Por ello, también se devuelve en los resultados una nueva columna que indique si cumple o no esa longitud establecida. Esto permite al usuario tener una visión más completa de la adecuación de los cebadores.

Desglose de los archivos y nueva información

Se modifican los resultados obtenidos hasta el momento para desglosarlos en nuevos archivos, así como añadir más información sobre la ejecución que pueda resultar de interés para el usuario en análisis propios posteriores.

Se genera un archivo CSV resumen de la ejecución, donde se muestran los cebadores analizados, asignándoles un identificador numérico para la relación de los datos con otros archivos, así como información sobre los parámetros introducidos por el usuario en la ejecución (*mismatches*, longitud mínima del fragmento, y demás). En este archivo resumen también aparece la cobertura, y es igual para ambos tipos de análisis (genes o genomas).

En otro fichero de tipo CSV más completo se pueden visualizar los fragmentos amplificados por cada par de cebadores para cada genoma/gen 16S, así como datos típicos de cada análisis (identificador, número de genes amplificados en un genoma, número de genes diferentes amplificados en un genoma, cantidad de *mismatches*, y más información interesante). Si los fragmentos amplificados encontrados superan los 3000pb, no se mostrará su secuencia.

Además de los resúmenes resulta interesante devolver archivos de tipo FASTA con los fragmentos amplificados por cada par de cebadores en los genomas/genes. Por ello, se decide crear para los genes 16S un archivo para cada par de cebadores en donde se recojan los fragmentos de los genes que serían amplificados y un archivo donde se recojan los genes que no contienen las secuencias diana compatibles para obtener una amplificación con este par de cebadores.

Como en el caso de los genomas la devolución de estos datos puede ser más caótica, se crea una carpeta para cada par de cebadores, y en ella se genera un archivo FASTA por cada genoma donde se guardan los fragmentos amplificados diferentes para dicho genoma. Que sean diferentes implica que se descartaron todos los duplicados a la hora de guardar los fragmentos amplificados en dicho archivo. Se empleará como cabecera de los fragmentos amplificados la cabecera del genoma junto con una cadena extra para diferenciarlos, ya que puede haber varios fragmentos amplificados en un mismo genoma. La cadena que se añade es `_V{Variant_No}`, donde

`Variant_No` se corresponde con el número de fragmento amplificado diferente del genoma, que se puede denominar variante.

Además, en esta carpeta se incluye un archivo de tipo CSV donde se relaciona cada archivo FASTA con el genoma analizado (el identificador GenBank indicado o bien el nombre del archivo del que se obtuvo) y si fue cubierto por el par de cebadores o no.

En el Anexo 9.2 se puede ver la estructura de los ficheros de resultados, su nomenclatura y su contenido.

Dos tipos de fuentes de genomas: CSV con identificadores GenBank o carpeta con FASTA

Se permiten al usuario dos posibles formas de analizar los pares de cebadores sobre genomas. Se permite indicar mediante línea de comandos un archivo CSV con los identificadores GenBank de los genomas sobre los que se quieran analizar los pares de cebadores o bien indicar una carpeta con los archivos FASTA donde se encuentren los genomas.

En el caso de especificar los genomas mediante los identificadores del GenBank, el programa obtendrá los genomas de la base de datos del NCBI mediante la herramienta Entrez del paquete BioPython. En el caso de indicar una carpeta con los archivos FASTA, se permitirá incluir más de un genoma por archivo.

Debido a estos cambios, se generan archivos nuevos para la realización de pruebas: carpetas con archivos FASTA de genomas y CSV con identificadores GenBank.

Depuración de los parámetros de entrada

Es necesario comprobar que los parámetros introducidos por el usuario sean válidos y realistas, por ellos se controlan las entradas y gestionan los errores de manera acorde.

Se controla el tamaño de los cebadores y su duplicidad. Si el tamaño de un cebador es inferior a 15pb, se muestra un aviso de tipo *warning*. Sin embargo, si el tamaño es inferior a 5pb se muestra un error y se detiene la ejecución del programa. Si una pareja de cebadores está duplicada, aparece un aviso de tipo *warning* y sólo se analiza una vez.

Se controla también la duplicidad en los genomas. Si aparece alguno de los identificadores GenBank repetidos se muestra un aviso y sólo se analiza una vez. Ocurre lo mismo en el caso de los genomas indicados mediante archivos FASTA, pero en este caso se aplica la función criptográfica *hash* sobre las secuencias de los genomas, ya que devuelve un valor único para

cada genoma, pudiendo comparar este valor en vez de toda la secuencia (más eficiente).

5. Pruebas y reflexión.

Se realizaron pruebas de la herramienta para comprobar el correcto funcionamiento después de cada tarea realizada. Estas pruebas a menor escala pretenden comprobar que la tarea se realizó correctamente y cumple con su objetivo. Así mismo, se procura verificar el buen funcionamiento general de la aplicación para descartar posibles fallos inducidos con la introducción de los cambios.

Al finalizar las tareas de implementación de la segunda fase se comienzan a realizar las pruebas generales y análisis de los resultados obtenidos. Estas pruebas pretenden asegurar el buen funcionamiento general de la aplicación y además comprobar la resiliencia del software ante posibles errores del usuario como, por ejemplo, parámetros de entrada incorrectos.

5.1 Pruebas del software

Se listarán a continuación las pruebas generales realizadas, indicando una breve descripción de lo que se realizó y si se superó o no.

Título	***Archivo CSV de cebadores con diferentes delimitadores***
Descripción	Se prueba a introducir el archivo CSV de cebadores con los delimitadores más comunes: coma, punto y coma.
Criterio de validación	Se deberán leer correctamente ambos archivos independientemente del delimitador.
Ejecución	Correcta.
Observaciones	No.

Título	***Archivo CSV de cebadores con cebadores repetidos***
Descripción	Se prueba a introducir el archivo CSV de cebadores con un par de cebadores repetido.
Criterio de validación	Se deberá detectar el cebador y lanzar un aviso por consola. Solo se analizará una vez.
Ejecución	Correcta.
Observaciones	No.

Título	***Archivo FASTA de genomas con genomas repetidos***
Descripción	Se prueba a introducir el archivo CSV de genomas con al menos un identificador GenBank repetido. Se realiza lo mismo pero con archivos FASTA de genomas.
Criterio de validación	Se deberán detectar los genomas repetidos en ambos casos, analizando sólo una vez dichos genomas.
Ejecución	Correcta.
Observaciones	No.

Título	***Archivos de entrada no existentes***
Descripción	Se prueba a introducir nombres de archivos no existentes para cualquiera de los archivos de entrada (cebadores, genes, genomas).
Criterio de validación	Se deberán detectar que los archivos no existen, lanzar un aviso en consola y detener la ejecución.
Ejecución	Correcta.
Observaciones	No.

Título	***Argumentos por línea de comandos insuficientes***
Descripción	Se prueba a introducir menos argumentos de los obligatorios.
Criterio de validación	Se deberán lanzar un aviso indicando qué parámetros obligatorios se deben introducir.
Ejecución	Correcta.
Observaciones	No.

Título	***Argumentos por línea de comandos inexistentes***
Descripción	Se prueba a introducir argumentos no existentes como parámetros de entrada.

Criterio de validación	Se deberán lanzar un aviso cada vez que se introduzca un argumento incorrecto.
Ejecución	Correcta.
Observaciones	No.

Título	***Número de mismatches fuera del rango***
Descripción	Se prueba a introducir un número de *mismatches* permitido fuera del rango establecido (entre 0 y 5)
Criterio de validación	Se deberán lanzar un aviso indicando el rango disponible para ese parámetro.
Ejecución	Correcta.
Observaciones	No.

Título	***Valores por defecto de los argumentos***
Descripción	Se prueba a no introducir ninguno de los argumentos opcionales para que se establezcan los valores por defecto.
Criterio de validación	Se deberá realizar el análisis con los valores establecidos por defecto en el software.
Ejecución	Correcta.
Observaciones	No.

Título	***Análisis de cebadores sobre genes***
Descripción	Se prueba a introducir como argumento genes para poder analizar los cebadores.
Criterio de validación	Se deberán generar dos archivos sobre la cobertura de los cebadores y dos archivos FASTA por par de cebadores con los genes cubiertos y no cubiertos.
Ejecución	Correcta.

Observaciones	No.

Título	***Análisis de cebadores sobre genomas***
Descripción	Se prueba a introducir como argumento genomas para poder analizar los cebadores.
Criterio de validación	Se deberán generar dos archivos sobre la cobertura de los cebadores y una carpeta por cada par de cebadores, donde habrá un archivo FASTA por cada genoma y un archivo CSV resumen de los genomas cubiertos y no cubiertos.
Ejecución	Correcta.
Observaciones	No.

Título	***Formato correcto en los archivos de resultados***
Descripción	Se generan los resultados de un análisis de cobertura de cebadores sobre genes 16S y genomas.
Criterio de validación	Los nombres de los archivos y de sus columnas (en caso de ser archivos CSV) deberán cumplir la nomenclatura establecida.
Ejecución	Incorrecta.
Observaciones	La nomenclatura de los nombres no se corresponde con la establecida.

Título	***Resultados correctos al finalizar el análisis sobre genes 16S***
Descripción	Se generan los resultados de un análisis de cobertura de cebadores sobre genes 16S.
Criterio de validación	Se buscan los pares de cebadores analizados manualmente sobre los genes empleando un editor de textos que admita realizar búsquedas con expresiones regulares. Se comprueba la exactitud de las columnas calculadas (longitud, *mismatches*…). Los resultados deberán ser coherentes.
Ejecución	Correcta.

Observaciones	No.

Título	***Resultados correctos al finalizar el análisis sobre genomas***
Descripción	Se generan los resultados de un análisis de cobertura de cebadores sobre genomas.
Criterio de validación	Al haber una gran cantidad de datos obtenidos de los genomas, se seleccionan aleatoriamente determinadas filas de los resultados para comprobar su validez. Se comprueba la exactitud de las columnas calculadas (longitud, *mismatches*…). Los resultados deberán ser coherentes.
Ejecución	Correcta.
Observaciones	No.

Título	***Generación de los logs con cada ejecución del programa***
Descripción	Se ejecuta múltiples veces el código para comprobar que se generan y actualizan los logs indicando los pasos y errores que se siguen en el código.
Criterio de validación	Deberá existir un archivo de logs por cada día en el que se ejecutó el software. Cada ejecución quedará registrada en el archivo del día en que se realizó.
Ejecución	Incorrecta.
Observaciones	Se determinó insuficiente la información mostrada en los logs, decidiendo indicar mejor los pasos que se realizaban en el código.

5.2 Análisis de los resultados obtenidos

A partir de las pruebas realizadas sobre el software se analizó la situación en la que se encontraba la aplicación. Se observaron qué pruebas fallaron y se determinó su gravedad.

De las pruebas realizadas, fallaron dos:

- Generación de los logs con cada ejecución del programa
- Formato correcto en los archivos de resultados

Se determinó que ambos fallos encontrados no eran graves ni costosos temporalmente, por lo que se procedió a implementar la solución.

Generación de los logs con cada ejecución del programa

Se complementa la información guardada en los logs:

- Se incluye la fecha exacta de guardado de una entrada en el log
- Se añade una entrada cuando finaliza el análisis.
- Se incluyen los errores relacionados con los argumentos de entrada
- Se incluye el progreso del programa: qué gen/genoma se está analizado en cada momento por un par de cebadores determinado

Formato correcto en los archivos de resultados

Se corrigen los nombres archivos generados que no cumplieran con la nomenclatura especificada en los anexos.

Se corrigen las columnas cuyo nombre no fuera representativo de su propósito. Por ejemplo, se modifica "*Primer No.*" a "*Primer Pair No.*", ya que es una referencia al par de cebadores que se está analizando.

6. Conclusiones

6.1 Lecciones aprendidas

Al finalizar este proyecto se ha obtenido un software de análisis de cobertura de cebadores analizados sobre genes 16S o genomas.

Durante su desarrollo se ha podido ver la utilidad, e incluso necesidad, de realizar una búsqueda de paquetes de Python ya existentes para suplir determinadas necesidades del software. La utilización de paquetes ya desarrollados permite ahorrar tiempo de implementación propia y dedicarlo a una mejora de la calidad y funcionalidad de la herramienta. Además, estos paquetes serán más estables y seguros en su ejecución que el código que se pueda realizar en el tiempo asignado para ello en el proyecto.

La implementación de un buen código permite adaptarse fácilmente a la introducción de nuevas funcionalidades o modificaciones sobre lo existente. Es importante desarrollar cualquier tipo de software pensando en los principios SOLID [33], que son cinco reglas básicas de programación para producir un software de calidad. Entre los objetivos que se recogen en estos principios se destacan:

- Software eficaz: debe ser robusto y estable.

- Código limpio y flexible ante nuevos cambios: debe ser reutilizable y mantenible.

- Software escalable: debe adaptarse fácilmente a la introducción de nuevas funcionalidades.

El diseño de cebadores es un proceso complejo, que puede llevar una gran cantidad de tiempo. Por ello, disponer de un software que permita probar su efectividad en términos de su cobertura aliviaría el coste del diseño de cebadores. Esta herramienta permite además mejorar iterativamente los cebadores diseñados a partir de los resultados detallados sobre su cobertura.

6.2 Objetivos

Al comienzo del proyecto se especificaron tres objetivos principales:

1. Analizar la cobertura de un conjunto de cebadores sobre un conjunto de genomas o genes 16S propuestos por el usuario a través de un software desarrollado en Python.

2. Permitir la configuración de la herramienta por parte del usuario, asegurando su usabilidad.

3. Crear un paquete de Python con la funcionalidad del software que permita su sencilla instalación en diversos entornos y sistemas operativos.

Los dos primeros objetivos se cumplieron satisfactoriamente. Se creó una herramienta que permite el análisis de la cobertura de cebadores sobre genes 16S o genomas. Es configurable a través de múltiples parámetros que permiten al usuario adaptarlo a las necesidades de su estudio. El software posee un comando de ayuda que indica al usuario qué parámetros puede introducir y sus posibles valores.

El último objetivo, consistente en la creación de un paquete en Python, se cumplió parcialmente. Se desarrolló el software teniendo en cuenta la posibilidad de convertirse en un paquete, pero no llega a ser el caso ya que no se publica en el repositorio de paquetes PyPi, y por tanto, no se puede descargar e instalar como tal. Esto se debe a que la funcionalidad de esta herramienta es ampliable, por lo que resultaría de mayor interés realizar más mejoras y ampliaciones de funcionalidad para complementar el software antes de su publicación.

6.3 Planificación y metodología

En rasgos generales, la planificación fue adecuada. Como en todo proyecto, se produjeron una serie de modificaciones sobre la planificación inicial al adaptarse ante los imprevistos surgidos durante la realización del proyecto.

En la primera fase se tuvo que realizar alguna tarea de configurabilidad de la aplicación mediante parámetros de entrada que inicialmente estaba pensada para realizarse en la segunda fase. Se tomó esta decisión para agilizar el desarrollo y proceso de pruebas del código a la hora de introducir datos de entrada. Por este mismo motivo, en esta misma fase también se incluyó la tarea de implementación de un *logger* que recoja el progreso, advertencias y errores de la ejecución del software.

En la segunda fase se incluyeron varias tareas más que no se habían contemplado inicialmente. A partir de las pruebas sencillas realizadas después de la implementación de la primera fase se encontraron algunas mejoras que se debían incluir en el software, que se corresponden con el apartado de mejoras de la funcionalidad.

La planificación del proyecto fue realizada desde el principio de un modo conservador, asegurando dejar un tiempo de margen antes de cada hito para poder adaptarse ante cualquier posible contratiempo, lo que permitió realizar todas las alteraciones antes comentadas sin verse comprometidas las entregas asociadas al TFM.

La situación del COVID-19 también fue una de las alteraciones a la que se vio sometido el proyecto, dificultando las reuniones con los consultores. Sin embargo, para paliar los efectos de esta situación, las fechas límites de entrega de los últimos hitos se aplazaron dos semanas.

6.4 Posibles mejoras

Durante el desarrollo del software surgieron múltiples ideas que se podrían incluir en el futuro para complementarlo.

Resultaría interesante realizar el análisis de cobertura también a nivel taxonómico. La herramienta devuelve la cobertura total de los pares de cebadores sobre todos los genomas o genes, es decir, para cada cebador se sabrá su nivel de cobertura general al evaluarlo sobre todas esas secuencias. Sin embargo, podría aportar gran cantidad de información devolver esa cobertura agrupando primero las secuencias que se están analizando según su taxonomía, en caso de analizar genomas. De ser así, se podría saber la cobertura concreta de un par de cebadores para niveles taxonómicos determinados.

En caso de llegar a publicarlo, sería beneficioso incluir unos archivos de entrada preestablecidos que, a través de un nuevo comando de la herramienta, se puedan ejecutar directamente y servir al usuario de ejemplo de uso.

De forma más opcional, se podría mostrar la información del progreso de la ejecución de manera más completa, incluyendo quizás una barra de progreso y una estimación del tiempo de ejecución restante que se actualizasen en todo momento, manteniendo al usuario informado.

7. Glosario

BLAST: programa informático de alineamiento de secuencias de tipo local, ya sea de ADN, ARN o de proteínas.

Cebador: en inglés, *primer*. Secuencia corta de ADN de cadena simple que se utiliza en una reacción en cadena de la polimerasa (PCR). En el método PCR se emplea un par de cebadores para hibridar con el ADN de la muestra y definir la región del ADN que será amplificada.

Cobertura: porcentaje de genomas o genes donde un par de cebadores es capaz de amplificar al menos un fragmento.

Deleción: tipo de mutación genética en la cual se pierde material genético, desde un solo par de nucleótidos de ADN hasta todo un fragmento de cromosoma.

Diagrama de clases: diagrama que describe la estructura de un sistema informático mostrando sus clases, relaciones entre éstas, atributos y operaciones o métodos.

Entrez: sistema de base de datos de biología molecular que proporciona acceso integrado a datos de secuencias de nucleótidos y proteínas, información de mapeo genómico y centrado en genes, datos de estructura 3D, PubMed MEDLINE y más.

Entrez Programming Utilities (E-Utilities): interfaz pública del sistema NCBI Entrez que permite el acceso a todas las bases de datos de Entrez, entre ellas PubMed, PMC, Gene, Nuccore (GenBank) y Protein.

Expresión regular: secuencia de caracteres que conforma un patrón de búsqueda.

FASTA: formato de fichero de texto que comienza con una descripción de una sola línea comenzando por el símbolo “>”, seguida de líneas de datos de secuencia.

Gap: mutación genética que se produce por inserciones o deleciones en la secuencia. Se suele representar un punto o un guion.

Gen 16S: gen que codifica el ARN ribosómico (ARNr) 16S, que es la macromolécula más ampliamente utilizada en los estudios de filogenia y taxonomía bacterianas.

GenBank: base de datos de secuencias genéticas del NIH, una colección de disponibilidad pública de secuencias de ADN.

Logger: herramienta de software para hacer un registro sistemático de eventos, observaciones o cualquier tipo de error que se produzca durante la ejecución de un programa.

Microsoft Project: software de administración de proyectos.

NCBI: es el Centro Nacional para la Información Biotecnológica, parte de la Biblioteca Nacional de Medicina de Estados Unidos una rama de los Institutos Nacionales de Salud (National Institutes of Health o NIH).

Notación IUPAC para las bases nucleotídicas: representación estándar de bases de ADN mediante caracteres individuales que especifican una sola base (por ejemplo, G para guanina, A para adenina) o un conjunto de bases (por ejemplo, R para G o A) definida por *International Union of Pure and Applied Chemistry* (IUPAC).

Primer: ver *cebador*.

PyPI: repositorio de software oficial para aplicaciones de terceros en el lenguaje de programación Python.

Python: lenguaje de programación interpretado, dinámico y multiplataforma.

TestPrime: herramienta que permite evaluar el rendimiento de los pares de cebadores ejecutando un PCR in silico en las bases de datos SILVA.

Visual Studio Code (VSCode): editor de código fuente que incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

8. Bibliografía

[1]. Weisburg WG, Barns SM, Pelletier DA, Lane DJ, 16S ribosomal DNA amplification for phylogenetic study, J Bacteriol, pp. 697-703, no. 2, vol. 173, 1991.

[2]. Case RJ, Boucher Y, Dahllöf I, Holmström C, Doolittle WF, Kjelleberg S, Use of 16S rRNA and rpoB genes as molecular markers for microbial ecology studies, Appl Environ Microbiol, pp. 278-288, no. 1, vol. 73, 2007.

[3]. Yarza P, Yilmaz P, Pruesse E, Glockner FO, Ludwig W, Schleifer KH, Whitman WB, et al, Nat Rev Microbiol, pp. 635-45, no. 9, vol. 12, 2014.

[4]. Sanger F, Coulson AR, A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase, J Mol Biol, pp. 441-448, no. 3, vol. 94, 1975.

[5]. Behjati S, Tarpey PS, What is next generation sequencing?, Arch Dis Child Educ Pract Ed, pp. 236-238, no. 6, vol. 98, 2013.

[6]. https://www.fiercebiotech.com/medical-devices/roche-to-close-454-life-sciences-as-it-reduces-gene-sequencing-focus 07/06/20

[7]. https://www.illumina.com/science/technology/next-generation-sequencing.html 07/06/20

[8]. Klindworth A, Pruesse E, Schweer T, et al., Evaluation of general 16S ribosomal RNA gene PCR primers for classical and next-generation sequencing-based diversity studies, Nucleic Acids Res, no. 1, vol. 41, 2013.

[9]. Mao DP, Zhou Q, Chen CY, Quan ZX, Coverage evaluation of universal bacterial primers using the metagenomic datasets, BMC Microbiol, no. 66, vol 12, 2012.

[10]. https://www.python.org/ 09/06/20

[11]. Benson DA, Cavanaugh M, Clark K, et al, GenBank. Nucleic Acids Res, 2013; 41(Database issue): D36-D42.

[12]. https://python-packaging.readthedocs.io/en/latest/dependencies.html 10/06/2020

[13]. https://pypi.org/ 09/06/20

[14]. https://www.microsoft.com/es-es/microsoft-365/project/project-management-software 08/06/2020

[15]. https://genome.ucsc.edu/goldenPath/help/iupac.html 07/06/20

[16]. https://www.genome.gov/es/genetics-glossary/Iniciador-o-cebador 07/06/20

[17]. Fanny Perraudeau, Sandrine Dudoit, James H. Bullard, bioRxiv, Accurate Determination of Bacterial Abundances in Human Metagenomes Using Full-length 16S Sequencing Reads

[18]. https://www.genome.gov/es/genetics-glossary/Delecion 08/06/20

[19]. Cock PJ, Antao T, Chang JT, et al., Biopython: freely available Python tools for computational molecular biology and bioinformatics, *Bioinformatics*, pp. 1422-1423, no. 11, vol. 25, 2009.

[20]. https://www.ncbi.nlm.nih.gov/ 08/06/20

[21]. https://docs.python.org/3/library/argparse.html 09/06/20

[22]. https://docs.python.org/3/library/os.html 09/06/20

[23]. https://pypi.org/project/regex/ 09/06/20

[24]. https://docs.python.org/3/library/csv.html 09/06/20

[25]. https://docs.python.org/3/library/logging.html 09/06/20

[26]. https://www.masqueteclas.com/articulo/el-concepto-de-clase/ 07/06/20

[27]. Isabel F. Escapa, Tsute Chen, Yanmei Huang, Prasad Gajare, Floyd E. Dewhirst, Katherine P. Lemon, mSystems, New Insights into Human Nostril Microbiome from the Expanded Human Oral Microbiome Database (eHOMD): a Resource for the Microbiome of the Human Aerodigestive Tract, no.6, vol. 3, 2018.

[28]. https://github.com/biopython/biopython/blob/master/Bio/SeqUtils/__init__.py 10/06/20

[29]. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ, Basic local alignment search tool, J Mol Biol, pp. 403-410, no. 3, vol. 215, 1990.

[30]. https://www.ncbi.nlm.nih.gov/books/NBK25501/ 08/06/20

[31]. https://www.ncbi.nlm.nih.gov/Web/Search/entrezfs.html 08/06/20

[32]. https://code.visualstudio.com/ 08/06/20

[33]. https://profile.es/blog/principios-solid-desarrollo-software-calidad/#los_principios_solid 18/06/20

9. Anexos

9.1 Manual de usuario

En este anexo se incluye el manual de usuario dónde se explica cómo utilizar la herramienta, detallando todas sus opciones y requisitos sobre los archivos de entrada.

Es requisito necesario disponer de Python 3.7 o una versión superior instalado. Como aún no estará disponible en un repositorio de descarga de paquetes para instalarlo automáticamente, habrá que ejecutar el archivo *setup.py* para instalar todas las dependencias del proyecto.

Una vez estén todos los paquetes necesarios listos, se podrá ejecutar el programa desde línea de comandos, accediendo a la ruta donde se encuentren los archivos del código.

Para ejecutar la herramienta, sin incluir ningún parámetro de entrada se indicará:

```
python3 ./main.py
```

Como no se indicaron ninguna de las opciones obligatorias (archivo con los cebadores y archivo con los genes/genomas), saltará un error y se indicará qué argumentos son obligatorios.

Para ver en detalle los requisitos de los argumentos se puede indicar la opción *--help* o *-h*.

```
python3 ./main.py -help
```

```
usage: main.py [-h] -gf file -pf file [-g choice] [-m number] [-mxs number]
               [-mns number] [-rd directory]

***--Gen 16S Primer Coverage--***

optional arguments:
  -h, --help            show this help message and exit
  -gf file, --sequence-file file
                        path to the CSV file with GenBank ids of genomes (will
                        download from NCBI), folder containing the FASTAs with
                        the genomes or FASTA with the genes to analyze
  -pf file, --primer-file file
                        path to the file with the primers to analyze, both
                        5'to 3'
  -g choice, --genome-gene choice
                        Type of sequences analyzed, genes or genomes
  -m number, --mismatches number
                        max number of mismatches allowed
  -mxs number, --max-size number
                        max size of amplified fragment
  -mns number, --min-size number
                        min size of amplified fragment
  -rd directory, --results-directory directory
                        results folder relative path (from current directory)
```

Ilustración 5: Ejecución del comando *help*

Se deberá disponer de dos archivos con los datos principales, que se indicarán al programa mediante:

- CSV para los cebadores (-pf o --*primer-file*): no tendrá cabecera, podrá disponer del delimitador que desee el usuario y se indicará un par de cebadores por fila.

- Archivo de genes o de genomas (-gf o --*sequence-file*):

    - FASTA con los genes 16S: tendrá la estructura normal de cualquier fasta.

    - CSV con los identificadores GenBank de los genomas: se indicarán cada uno separado en una línea diferente o en una misma línea separado por comas.

    - Carpeta con archivos FASTA que contengan los genomas: tendrán la estructura normal de cualquier fasta.

- Tipo de análisis (-g o --*genome-gene*): *genes* o *genomes* en función de las secuencias sobre las que se analice la cobertura. Por defecto será *genomes*.

Se dispondrá de archivos de ejemplo en la carpeta denominada *test_files*.

Comando para obtener un análisis base sobre genes:

```
python ./main.py -g genes -pf test_files/primers.csv -gf test_files/genes.fasta
```

Comando para obtener un análisis base sobre genomas a partir de un archivo CSV:

```
python ./main.py -g genomes -pf test_files/primers.csv -gf test_files/genomes.csv
```

Comando para obtener un análisis base sobre genomas a partir de una carpeta con archivos FASTA:

```
python ./main.py -g genomes -pf test_files/primers.csv -gf test_files/genomes
```

En cualquiera de los casos anteriores se generará una carpeta denominada results donde se guardan todos los resultados del análisis. Será diferente según el tipo de análisis y se explicará en detalle en el Anexo 9.2.

Durante la ejecución se irá mostrando el progreso del análisis cebador a cebador, según se vaya comprobando su cobertura para cada gen/genoma.

Se dispone también de múltiples comandos extra opcionales que aparecen listados al utilizar el comando *help* y se explican a continuación:

- Máxima cantidad de *mismatches* admitidos (-m o *–mismatches*): por defecto será 0 y como máximo 5.

- Tamaño máximo del fragmento amplificado (-mxs o *–max-size*): por defecto será 600pb.

- Tamaño mínimo del fragmento amplificado (-mns o *–min-size*): por defecto será 200pb.

- Ruta a la carpeta donde se guardarán los resultados (-rd o *–results-directory*): por defecto se creará una carpeta llamada *results* en el directorio desde el que se ejecute el comando.

9.2 Estructura de archivos de resultados

En este anexo se explica la estructura de los archivos de resultados generados por la herramienta, así como la nomenclatura de estos.

Dependiendo del tipo de análisis, genes o genomas, se generarán archivos distintos.

Genes 16S

En el caso del análisis de cobertura de cebadores sobre genes, al ser más sencillo, se generarán menos archivos y tendrán menos cantidad de información.
coverage_summary.csv

Archivo que aporta un resumen del análisis con los datos más relevantes de cada par de cebadores, como la propia información introducida por el usuario y la cobertura.

COLUMNA	DESCRIPCIÓN
PRIMER PAIR NO	Identificador del par de cebadores. Se corresponde con el orden en el que se encuentren en el archivo.
F PRIMER	Secuencia del cebador directo
R PRIMER	Secuencia del cebador inverso
MISMATCHES	Número de *mismatches* máximo
MIN. FRAGMENT SIZE	Tamaño máximo del fragmento

MAX. FRAGMENT SIZE	Tamaño mínimo del fragmento
COVERAGE	Cobertura del par de cebadores
ANALYZED GENES	Número de genes analizados
AMPLIFIED GENES	Número de genes amplificados por el par de cebadores
NON-AMPLIFIED GENES	Número de genes no amplificados por el par de cebadores

coverage.csv

Archivo que aporta información sobre los fragmentos amplificados por cada par de cebadores

COLUMNA	DESCRIPCIÓN
PRIMER PAIR NO	Identificador del par de cebadores. Se corresponde con el orden en el que se encuentren en el archivo.
AMPLIFIED GENE	Identificador del gen 16S donde se encontró el fragmento amplificado. Se corresponde con el orden en el que se encuentren en el archivo.
DISTANCE	Tamaño del fragmento amplificado
DISTANCEOK	Valor booleano (Yes o No) según el tamaño del fragmento amplificado se encuentre en el rango establecido por el usuario.
MISMATCHES	Número de *mismatches* en el fragmento
AMPLIFIED SEQUENCE	Secuencia del fragmento

{Primer_No}_Amplified.fasta

Archivo FASTA que contiene los genes 16S indicados por el usuario donde se encontró un fragmento amplificado por un par de cebadores determinado. Se generan tantos archivos como pares de cebadores analizados, donde *Primer_No* se corresponde con el identificador del par de cebadores.

{Primer_No}_Non_Amplified.fasta

Archivo FASTA que contiene los genes 16S indicados por el usuario donde no se encontró ningún fragmento amplificado por un par de cebadores determinado. Se generan tantos archivos como pares de cebadores analizados, donde *Primer_No* se corresponde con el identificador del par de cebadores.

Genomas

En el caso del análisis de cobertura de cebadores sobre genomas se genera mayor cantidad de información que también es más compleja. Por ello, se incluirán carpetas en la estructura de los resultados para organizar mejor los datos.

coverage_summary.csv

Similar al de los genes 16S, archivo que aporta un resumen del análisis con los datos más relevantes de cada par de cebadores, como la propia información introducida por el usuario y la cobertura.

COLUMNA	DESCRIPCIÓN
PRIMER PAIR NO	Identificador del par de cebadores. Se corresponde con el orden en el que se encuentren en el archivo.
F PRIMER	Secuencia del cebador directo
R PRIMER	Secuencia del cebador inverso
MISMATCHES	Número de *mismatches* máximo
MIN. FRAGMENT SIZE	Tamaño máximo del fragmento
MAX. FRAGMENT SIZE	Tamaño mínimo del fragmento
COVERAGE	Cobertura del par de cebadores
ANALYZED GENOMES	Número de genomas analizados
COVERED GENOMES	Número de genomas cubiertos por el par de cebadores
NON-COVERED GENOMES	Número de genomas no cubiertos por el par de cebadores

coverage.csv

Archivo que aporta información sobre todos los fragmentos amplificados por cada par de cebadores en cada genoma. No se incluirá la secuencia de aquellos fragmentos que superen 3000pb.

COLUMNA	DESCRIPCIÓN
PRIMER PAIR NO	Identificador del par de cebadores. Se corresponde con el orden en el que se encuentren en el archivo.
GENOME	Identificador del genoma donde se encontró el fragmento amplificado. Se corresponde el identificador GenBank o con

	el nombre del fichero FASTA, según el tipo de entrada.
STRAND	Hebra del genoma en la que se encuentra el fragmento
INITIAL POSITION FRAGMENT	Posición en el genoma donde comienza el fragmento
END POSITION FRAGMENT	Posición en el genoma donde termina el fragmento
DISTANCE	Tamaño del fragmento amplificado
DISTANCEOK	Valor booleano (Y o N) según el tamaño del fragmento amplificado se encuentre en el rango establecido por el usuario.
MISMATCHES	Número de *mismatches* en el fragmento
AMPLIFIED GENES	Número de fragmentos (genes) encontrados en el genoma
UNIQUE GENES	Número de fragmentos (genes) diferentes encontrados en el genoma. Se descartan los duplicados para obtener esta cifra.
AMPLIFIED GENE	Secuencia del fragmento

primer_{Primer_No}

Carpeta que contendrá los archivos FASTA con todos los fragmentos amplificados de cada genoma. Habrá un archivo de este tipo por genoma cubierto, y su nombre será `{Genome_Id}_covered.fasta`.

Asimismo, también se guarda un archivo denominado `analyzed_genomes.csv`, donde aparecerá un resumen de los genomas analizados por cada par de cebadores. Para cada genoma cubierto se indicará el nombre del archivo FASTA asociado con los fragmentos amplificados.

COLUMNA	DESCRIPCIÓN
GENOME	Identificador del genoma donde se encontró el fragmento amplificado. Se corresponde el identificador GenBank o con el nombre del fichero FASTA, según el tipo de entrada.
AMPLIFIED	Valor booleano (Y o N) según el genoma tiene fragmentos amplificados o no.
SEQUENCE FILENAME	Nombre del archivo de fragmentos amplificados correspondiente al genoma