UNIVERSIDAD
NACIONAL
DE COLOMBIA

Identificación de *contigs* asociados a plásmidos obtenidos a partir de secuenciación de genoma completo de aislamientos de *Klebsiella pneumoniae.*

Diego Camilo Talero Osorio

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento Ingeniería de Sistemas e Industrial
Bogotá D.C, Colombia
2022

Identificación de *contigs* asociados a plásmidos obtenidos a partir de secuenciación de genoma completo de aislamientos de *Klebsiella pneumoniae.*

Diego Camilo Talero Osorio

Trabajo de investigación presentado como requisito parcial para optar al título de:

Magister en Bioinformática

Director:

Ph.D Emiliano Barreto Hernández

Línea de Investigación:
Diagnóstico molecular
Grupo de Investigación:
Bioinformática

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento Ingeniería de Sistemas e Industrial
Bogotá D.C, Colombia
2022

Dedicatoria

Dedicado a todo aquel, que encuentre en estas páginas, la oportunidad, de algo nuevo aprender.

Declaración de obra original

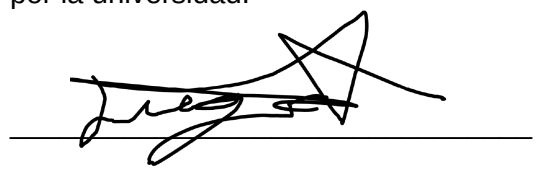
Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.



Nombre

Fecha 8/08/2022

Agradecimientos

Agradezco en primer lugar a mi madre a quien con su constante apoyo y amor, brindó la fortaleza para continuar este camino sin desfallecer, a mi padre por sus enseñanzas que han sido herramientas para enfrentar las adversidades, a mi tutor Emiliano Barreto Hernández quien permitió y guio esta fascinante posibilidad de aprender, a mis compañeros del grupo de investigación de Bioinformática del Instituto de Biotecnología Sebastián Prada, Mishelle Cuello, Caridad Tenorio, Laura Rojas, Nicole Osorio y Karen Aguilar; a quienes con su amistad y diversas áreas de conocimiento han nutrido y motivado a mi persona, y al apoyo de Colciencias que ha permitido la exitosa ejecución de este trabajo.

Resumen

Identificación de *contigs* asociados a plásmidos obtenidos a partir de secuenciación de genoma completo de aislamientos de *Klebsiella pneumoniae.*

Uno de los problemas frecuentes en salud pública son las Infecciones Asociadas a la Atención en Salud (IAAS), La Organización Mundial de la Salud (WHO) ha publicado una lista de microorganismos de prioridad clínica (WHO, 2017), entre los cuales a nivel crítico están todas las *Enterobacterias* que presentan resistencia a antibióticos carbapenémicos como *Klebsiella pneumoniae* que suele contar con múltiples mecanismos de resistencia frente a dichos antibióticos (Schroeder, Brooks, & Brooks, 2017). El desarrollo de tecnologías de secuenciación de nueva generación (NGS) ha permitido el estudio del "comportamiento" y /o "composición" de los genomas de microorganismos de interés clínico; así mismo también se han diseñado y desarrollado algoritmos y flujos de trabajo bioinformáticos para el almacenamiento, anotación y análisis de estos datos, que han facilitado identificar y caracterizar, un gran número de elementos genómicos involucrados en los mecanismos de resistencia. En este trabajo se propone una herramienta de clasificación de *contigs* pertenecientes a plásmidos, obtenidos por secuenciación de genoma completo (WGS), que implementa varias de las herramientas, que a través de un método experimental iterativo fueron configuradas para obtener un rendimiento maximizado para las cepas de trabajo de *K. pneumoniae*.

Palabras clave: PipeLine, algoritmo, *Klebsiella pneumoniae*, Plásmidos, Secuenciación de Nueva Generación, Lecturas, *Contigs*.

Abstract

Identification of *contigs* associated with plasmids obtained from whole-genome sequencing of *Klebsiella pneumoniae* isolates.

One of the frequent problems in public health is the Infections Associated with Health Care (IAAS). The World Health Organization (WHO) published a list of microorganisms of clinical priority (WHO, 2017), among which at the critical level are all Entero-bacteria with resistance to carbapenems like *Klebsiella pneumoniae*, which usually has several mechanisms of resistance (González Rocha et al., 2017), frequently associated with the genetic information (Schroeder et al., 2017). The development of New Generation Sequencing technologies (NGS) allows the study of the "behavior" and/or "composition" of the microorganism genomes of clinical interest. Likewise, algorithms and bioinformatics workflows have been designed and developed for the storage, annotation, and analysis of these data, to the point of identifying and characterizing a large number of genomic elements involved in resistance mechanisms. This work shows the implementation of a *contig* classification pipeline designed to choose which of them are part of a plasmid. It uses *contigs* obtained by NGS technologies and implements several programs to carry out this task, which, thanks to an iterative experimental method, were configured to obtain a maximized yield for the working strains of K. pneumoniae.

Keywords: PipeLine, algorithm, *Klebsiella pneumoniae*, Plasmids, New Generation Sequencing (NGS), *reads*, *contigs*, in silico

Este Trabajo Final de maestría fue calificado en junio de 2022 por el siguiente evaluador:

Andrés Mauricio Pinzón Velasco PhD.
Profesor Instituto de Genética

Universidad Nacional de Colombia

Contenido

Pág.

Lista de figuras

Pág.

Tabla de Ecuaciones

Lista de tablas

Pág.

Abreviaturas

Abreviatura	Término
IAAS	Infecciones Asociadas a Atención en Salud
OMS	Organización Mundial de la Salud
NGS	*Next generation Secuencing*
WGS	*Whole Genome Sequencing*
ASCII	*American Standard Code for Information Interchange*
FASTQ	Texto plano para almacenar información biológica con calidad específica a cada Nucleótido
SGD	Descenso de Gradientes Estocástico
SVM	*Suport Vector Machine*
K-mer	Sub-secuencia de nucleótidos de tamaño K
WGM	*Whole Genome Mapping*
AUC	*Area Under the Curve*
SRA	*Sequence Read Archive*

Introducción

Desde que la humanidad tiene la capacidad de combatir infecciones bacterianas a través de los antibióticos, se ha iniciado un proceso de evolución mutua entre las bacterias y el diseño de nuevos antimicrobianos. Las bacterias cambian sus mecanismos de resistencia fácilmente, ya que pueden mutar o utilizar mecanismos de transferencia horizontal de genes, que les permiten adoptar nuevas herramientas de supervivencia frente a los antimicrobianos (Martinez, Coque, & Baquero, 2015). En salud pública existe un problema relacionado con las infecciones asociadas a la atención en salud (IAAS), ya que estas pueden generar infecciones de implantes, fibrosis quística, infecciones de tracto urinario y enfermedades periodontales. Estos microorganismos pueden trabajar en conjunto formando estructuras heterogéneas o biopelículas bacterianas que les brindan entre 100 a 1000 veces más tolerancia a los antimicrobianos (Olsen, 2015). A medida que las bacterias adquieren resistencia a los antibióticos convencionales, se magnífica entonces su peligrosidad y alcance, siendo así un desafío el desarrollo de nuevos antibióticos (X. Z. Li, Plesiat, & Nikaido, 2015).

La situación representa hoy en día uno de los mayores riesgos de salud pública a nivel mundial, por lo que para la OMS (Organización Mundial de la Salud) es una prioridad de nivel crítico tener herramientas más eficientes y oportunas que ayuden a identificar y caracterizar los microorganismos peligrosos para la salud pública a un nivel global (WHO, 2017).

Este reto se aborda inicialmente identificando los elementos genéticos que confieren esta resistencia al microorganismo, para lo cual la secuenciación de nueva generación, también llamada de alto rendimiento o (NGS) por sus siglas en inglés "*Next generation Secuencing*", hace posible en un tiempo relativamente corto obtener la secuencia de ADN genómico de un microorganismo (Bootsma & Schouls, 2015). El ADN codifica la información necesaria para el funcionamiento celular, en este caso de las bacterias, información dentro de la cual están también sus mecanismos de defensa, codificados por genes presentes en el

cromosoma bacteriano o en plásmidos, estos últimos obtenidos por transferencia horizontal, mecanismo utilizado por las bacterias que les permite intercambiar material genético que en general las hace más resistentes a las condiciones adversas del medio ambiente donde viven, confiriéndoles por ejemplo, resistencia a los antibióticos o haciéndolas más virulentas (Blair, Webber, Baylay, Ogbolu, & Piddock, 2015). Cabe resaltar la alarmante aparición de bacterias que cuentan con múltiples plásmidos, los cuales brindan distintas características fenotípicas para diseminarse y resistir el ataque de los antibióticos (Liu et al., 2016).

En el presente trabajo final de Maestría el objetivo del primer capítulo es describir el funcionamiento y características de los plásmidos, así como la importancia de identificarlos. Además, se muestra cómo es que, gracias a las novedosas metodologías NGS de fragmentos cortos como la desarrollada por Illumina, se obtienen simultáneamente fragmentos de secuencia (*contigs*) de cromosoma y plásmidos de una bacteria, que por efecto del proceso resultan mezclados, en un único archivo sin conocimiento del componente genómico asociados. También se hace una descripción de algunas herramientas bioinformáticas disponibles para resolver el problema de diferenciar las secuencias correspondientes a los plásmidos, a partir de datos WGS obtenidos por tecnologías NGS, y por ultimo una justificación de la necesidad y la aproximación propuesta en el presente trabajo.

En el segundo capítulo se describe a detalle el trabajo realizado para el diseño y configuración del pipeline, propuesto para resolver la tarea de clasificación de secuencias plasmídicas y cromosomales a partir de la información de genoma completo de cepas de *K. pneumoniae*, obtenidas por tecnología Ilumina Hiseq 2500. Este flujo de trabajo escrito en lenguaje Python 3.7, involucra cinco herramientas de identificación de secuencias de plásmidos:

1. PlasmidSPAdes (Antipov, Korobeynikov, McLean, & Pevzner, 2016).
2. PlasFlow (Pawel S Krawczyk, Lipinski, & Dziembowski, 2018).
3. Alineamiento de los *contigs* contra una base de datos especifica de cromosomas de *K. pneumoniae*.
4. Alineamiento de los *contigs* contra una base de datos especifica de secuencias plásmidicas de Bacterias reportadas y anotadas en el NCBI.
5. Filtrado por variación de profundad de los *contigs* ensamblados con el programa Spades.

1. Capítulo

Marco Teórico

Naturaleza Bacteriana

En el momento que se diseña un plan de contingencia para combatir un microorganismo, es importante conocer desde la genómica, los mecanismos de defensa, cómo se expresan y cómo se controlan; en el caso de las bacterias se tienen dos grupos de secuencias genómicas: las cromosomales y las plasmídicas Figura 1. Las primeras encargadas de todas las funciones que tienen que ver con la vida celular, y las segundas que les atribuyen propiedades específicas para determinados ambientes. Estas secuencias se encuentran organizada en genes que realizan y regulan la expresión de mecanismos propios de las bacterias patógenas, como es la colonización de tejidos, invadir o producir toxinas que causan los síntomas de la enfermedad infecciosa (Founou, Founou, & Essack, 2018).

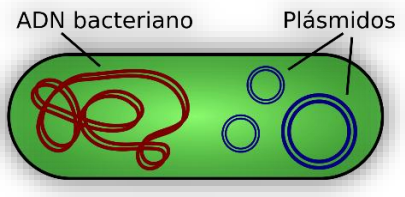


Figura 1. (Bacteria, esquema genético básico)

Los plásmidos son secuencias pequeñas de ADN, cuya estructura es circular tienen una frecuencia de replicación distinta a la del cromosoma, lo que conlleva a que haya más de una copia por ciclo celular. Los plásmidos se pueden clasificar por su tamaño, número de copias o el tipo de genes (virulencia, resistencia a antibióticos), y por grupos de

incompatibilidad; los plásmidos que pertenecen al mismo grupo de incompatibilidad son aquellos que pueden coexistir en la misma bacteria (Ross & Schultz, 1996).

Para analizar los plásmidos es primordial separarlos del resto de secuencias genómicas (Alimolaei & Golchin, 2017). Esta tarea puede realizarse físicamente en el laboratorio usando técnicas como amplificación de replicones o de genes de partición utilizando PCR y luego secuenciación de los fragmentos de ADN obtenidos (Bousquet et al., 2015). El problema es que se obtienen secuencias parciales de los plásmidos basadas en los iniciadores o *primers* utilizados durante la PCR. Para evitar esto, actualmente se utilizan las metodologías de secuenciación de nueva generación (NGS), que permiten la secuenciación de genoma completo (WGS) de la bacteria, proporcionando la secuencia tanto del cromosoma como de los plásmidos presentes. Para procesar y analizar *in silico* los resultados de la WGS, se han implementado herramientas bioinformáticas que permiten discriminar entre las secuencias del cromosoma y las de los plásmidos que puedan estar presentes, a igual que hacer comparaciones e identificar las funciones biológicas de las secuencias obtenidas.

Secuenciación de nueva generación

Los procesos bioinformáticos comienzan con la necesidad de conocer la información contenida en el ADN y Desde 1869 cuando se aisló por primera vez el ADN por el biólogo *Frierich Miescher*, numerosos esfuerzos se han llevado a cabo para descifrar de forma cada vez más precisa la manera en que la información contenida en estas moléculas orgánicas funciona. La NGS es un concepto que engloba las tecnologías encargadas de obtener de manera masiva la secuencia de ácidos nucleicos contenidas en un organismo. En la Tabla 1, se pueden apreciar las características principales de los métodos de secuenciación disponibles, desde Sanger en la primera generación, los métodos *single end* Ion Torrent y 454 Roche, los de menor tasa de error, bajo costo y más implementados ofrecidos por Illumina, y los de tercera generación *PacBio y Oxford Nanopore.*

Tabla 1. Métodos de Secuenciación (Rubio, Pacheco-Orozco, Gómez, Perdomo, & García-Robles, 2020)

Method	Generation	Read length (bp)	Single pass error rate (%)	No. of reads per run	Time per run	Cost per million bases (USD)
Sanger ABI 3730×l	1st	600–1000	0.001	96	0.5–3 h	500
Ion Torrent	2nd	200	1	8.2×10^7	2–4 h	0.1
454 (Roche) GS FLX +	2nd	700	1	1×10^6	23 h	8.57
Illumina HiSeq 2500 (High Output)	2nd	2 × 125	0.1	8×10^9 (paired)	7–60 h	0.03
Illumina HiSeq 2500 (Rapid Run)	2nd	2 × 250	0.1	1.2×10^9 (paired)	1–6 days	0.04
SOLiD 5500×l	2nd	2 × 60	5	8×10^8	6 days	0.11
PacBio RS II: P6-C4	3rd	$1.0–1.5 \times 10^4$ on average	13	$3.5–7.5 \times 10^4$	0.5–4 h	0.40–0.80
Oxford Nanopore MinION	3rd	$2–5 \times 10^3$ on average	38	$1.1–4.7 \times 10^4$	50 h	6.44–17.90

La tecnología *Illumina,* sobre la cual se centran los análisis del presente trabajo, es descrita en los siguientes pasos generales:

- Se aísla el ADN de la materia celular.
- El ADN es fragmentado con esferas, por enzimas o por procesos mecánicos de forma aleatoria.
- Se añaden adaptadores y marcadores a cada fragmento para su posterior identificación en la celda de flujo.
- Estos fragmentos se amplifican generando agrupaciones o *clusters* de la misma secuencia dentro de cada *tile* de la celda de flujo.
- Cada *cluster* será entonces un punto de censado y secuenciado, produciendo los *reads forward* y *reverse* correspondientes a las secuencias terminales de cada fragmento.

El producto de una secuenciación son los *reads*, los cuales corresponden a las secuencias obtenidas a partir del genoma en estudio. Su tamaño depende de la metodología y librerías implementadas ver Tabla1, siendo reportados en archivos en distintos formatos, dentro de los cuales uno de los más utilizados actualmente es el *FASTQ,* el cual contiene la secuencia de nucleótidos con su respectivo valor de calidad codificado en caracteres ASCI. Para el caso de un genoma bacteriano este conjunto de *reads* contiene una mezcla de los que se derivan del cromosoma y los derivados de los plásmidos presentes en la bacteria al momento de secuenciar.

Ensamble *de novo*

El ADN secuenciado resulta en millones de fragmentos que deben ser organizados para obtener la secuencia de nucleótidos del genoma, lo cual se realiza implementando el procesamiento *in silico* de estos pequeños fragmentos, donde se procede a agruparlos en conjuntos de *reads* que se sobreponen. de estos alineamientos múltiples se obtienen las secuencias consenso a las que se denominan *contigs* (Chaisson, Pevzner, & Tang, 2004), Las principales metodologías para realizar el Ensamble *de novo* son (Aguilar-Bultet & Falquet, 2015):

- **Greedy**, Algoritmo de búsqueda de rutas, que traza un camino sobre un grafo que representa el alineamiento. En este grafo cada r*ead se* representan como un nodo y la similitud entre *reads* crean las aristas que conectan los nodos, esta propuesta no suele explorar todo el espacio de búsqueda, en vez, avanza tomando decisiones secuenciales que suelen llevarlo a un óptimo local.
- **Overlap Layout Consensus (OLC)**, Es una metodología que resuelve esta tarea utilizando grafos, donde los nodos son las lecturas y las relaciones entre ellos representan que tantos nucleótidos comparten. El algoritmo busca el camino que mejor resuelva el grafo, es decir la ruta más corta y con mayor cantidad de nucleótidos compartidos entre las lecturas.
- **Gráficos De Bruijn**, Esta metodología implementa también grafos para resolver la continuidad de las secuencias genéticas, pero a diferencia del método OLC que usa la semejanza entre lecturas como conector, esta usa para un grafo sub-secuencias de tamaño impar especifico llamados *k-mers*, que representan la relación entre las lecturas solo si dos lecturas comparten el mismo *k-mer*. Esta metodología es la base de las herramientas de Ensamble *de novo,* más populares actualmente, que buscan la secuencia genómica que mejor resuelva el grafo, secuencia que varía ampliamente según los tamaños de *k-mer* utilizados.

Selección del tamaño de *K-mer* para ensamble *de novo*

En la ejecución de un ensamble *de novo* los resultados son sensibles a los tamaños de *k-mer* seleccionados (D. Li, Liu, Luo, Sadakane, & Lam, 2015).

- Los tamaños de *k-mer* **pequeños** pueden ser útiles en zonas de baja cobertura pues la cantidad de información es reducida y es posible que con la fragmentación que produce un corto tamaño se pueda resolver la secuencia contigua, a diferencia de una zona con alta cobertura donde puede generar un exceso de rutas y por ende un grafo más complejo.
- Los tamaños ***grandes*** de *k-mer* suelen ser los más recomendados, ya que por su tamaño es más probable el sobrelapamiento, lo que generará zonas contiguas más largas, que resultaran en un Ensamble *de novo* más extenso.

Es usual ver en la comunidad de bioinformática que este desafío de la selección de *k-mer* en el ensamble *de novo* es abordado tanto en transcriptómica (Durai & Schulz, 2016) como en genómica, seleccionando al azar unos tamaños no mayores al tamaño de la lectura, ejecutando varios Ensambles y evaluándolos, para de esta manera seleccionar aquel que se desempeña mejor en métricas como: cobertura del genoma esperado, NG50, numero de *contigs* y cantidad de errores; o además de las métricas usuales, emplear el mapeo contra un genoma (WGM), para evaluar los Ensambles de diferentes tamaños de *k-mer* en función de los *misassemblies* (Xavier et al., 2014). Para mejorar la selección de los *k-mer*s a utilizar en las herramientas de Ensamble *de novo*, se han desarrollado herramientas de conteo de *k-mer* y selección de *k-mer*, como *jellyfish* (Marçais & Kingsford, 2011) y *kmergenie* (Chikhi & Medvedev, 2014) respectivamente. La primera permite al usuario realizar un conteo de *k-mer* específico para ciertos tamaños ingresados y la segunda funciona basada en histogramas de conteo de *k-mer* y aproximaciones heurísticas, para hacer un cálculo del tamaño de *k-mer* que produce la mayor cantidad de secuencias de tamaño k distintas, que como consecuencia esperada generará un Ensamble genómico más extenso. Herramientas como *Spades* (Xavier et al., 2014), que ensambla apoyándose en distintos grafos de *bruijn*, también permite implementar un rango de *k-mer* entre 21 y 127, esto no indica que el Ensamble será mejor o peor, pues la correcta selección de estos sigue siendo una responsabilidad del bioinformático. Los autores de esta herramienta multi-kmer recomiendan para lecturas mayores de 250 pb utilizar 6 *k-mers* que cubran el espectro de posibilidades dese 21 hasta 127 pb con los valores (21,33,55,77,99,127). Esta elección de *k-mer*s da buenos resultados en la mayoría de los casos, pero no asegura que genere el mejor ensamble (Xavier et al., 2014).

En adición a los desafíos de elección de *k-mer* un problema tangible para la secuenciación de alto rendimiento es que el corto tamaño de los *reads* genera en el ensamble *de novo* sensibilidad a las zonas repetitivas de un genoma, pues estas son más grandes que los tamaños de las lecturas y aunque se seleccione un tamaño de *k-mer* tan grande como el *read*, no es posible dar continuidad a estas zonas repetitivas resultando en un Ensamble fragmentado.

Estado del Arte

Identificación *in silico* de plásmidos a partir de secuenciación de genoma completo (WGS)

La bioinformática se basa en la sinergia entre biología, estadística, matemáticas e informática para resolver problemas de tratamiento y análisis de la información biológica, dentro de los que se encuentra el discernir a partir de los resultados de WSG de bacterias, cuales secuencias son de plásmidos y cuáles del cromosoma. Para la identificación de plásmidos desde la secuencia del genoma completo de un microorganismo existen varias herramientas bioinformáticas disponibles, las cuales serán presentadas a continuación.

PlaScope

Esta herramienta consta de un pipeline que implementa una metodología de clasificación por comparación con la herramienta Centrifuga, desarrollada para clasificación de secuencias en metagenómica; para la clasificación de plásmidos PlaScope implementa una base de datos conformada por todas las secuencias de cromosomas y plásmidos de *E. coli* disponibles en el NCBI (Genome) (Van Horn, Wu, Zheng, Dai, & Chen, 2019) para el 10 de enero del año 2018. **Centrifuga** es una herramienta que implementa el método de comparación como lo hace también Blast, pero usa un método de indexación de la base de datos *Target* (secuencias objetivo contra las que se comparan los elementos desconocidos o no clasificados) y el *Subject* (secuencia problema). Esta indexación creada por Burrows Wheeler y Ferragina Monzini (Royer et al., 2018), reduce el esfuerzo computacional de procesamiento de grandes cantidades de información, permitiendo manejar una mayor cantidad de información en un tiempo relativamente más corto (Kim, Song, Breitwieser, & Salzberg, 2016). ***PlaScope*** retorna las secuencias tipo *contig* que

como resultado de la comparación por centrifuga tienen una semejanza lo suficientemente concluyente como para determinar que aquella secuencia forma parte de un plásmido.

PlasFlow

Implementa una base de datos de genomas bacterianos obtenida del NCBI, y como herramienta de clasificación implementa inteligencia artificial discriminando los plásmidos por un modelo que evalúa el contenido de *k-mers*. Esta evaluación parte de la extracción de características de cada elemento en el set de datos, en este caso los *k-mer*s de tamaño constante y sus frecuencias, lo que en inteligencia artificial es comúnmente llamado "*Bag of words*". Esto permite hacer una extracción de información transversal a todos los elementos, que posteriormente permitirá al modelo comparar el patrón problema con los aprendidos en el proceso de entrenamiento. Ya que los plásmidos contienen porcentajes de (GC) diferentes a los cromosomales (Nishida, 2012), esta característica tiene un efecto directo en el contenido de *k-mer* de los *contigs* como se evidencia en la

Figura 2, donde se muestra que esta diferencia es relativa a la del cromosoma, y depende también de la especie.

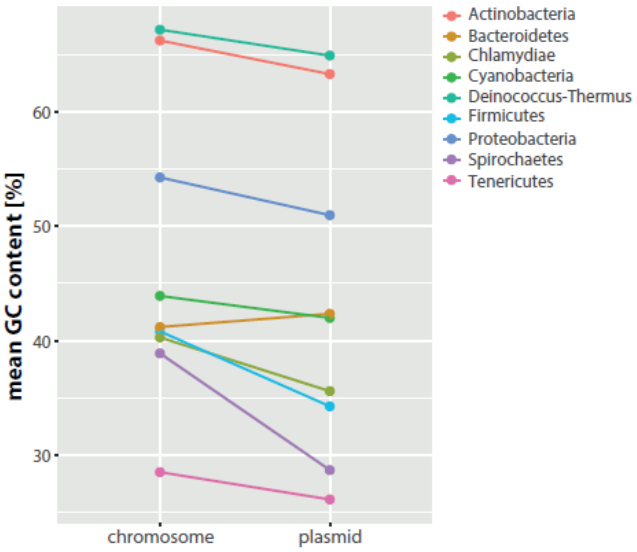


Figura 2. Relación de %GC del cromosoma en comparación con plásmidos para diferentes tipos de bacterias.

Blast

Es una herramienta común para muchos proyectos bioinformáticos, ya que está diseñada para comparar localmente una secuencia *query* con numerosas secuencias *target* de una base de datos. Como es el caso de Gem-Pro (Torres Manno et al., 2019), una herramienta diseñada para la identificación de genes cercanos implementando el alineamiento del software *Blast*. El algoritmo que la implementa ha evolucionado, ya que a través del tiempo las bases de datos aumentan su cantidad de información de manera exponencial cada año, lo que genera la necesidad de mejorar la eficiencia con la que el programa realiza las comparaciones. Este proceso lo realiza en tres pasos principales (Program & Guide, 2009):

- **Obtener los *seeds***: De la secuencia *query* Blast extrae secuencias cortas con características estadísticamente conservadas llamadas “seed” (semillas), esto para disminuir el costo computacional de comparar toda la secuencia *query*.
- ***Seeding***: Blast busca de manera heurística, lo que significa que utilizando las semillas del *query* encuentra las secuencias de la base de datos que contienen uno o varios de estos fragmentos, contra las que posteriormente vale la pena hacer una comparación completa de sus secuencias.
- **Extensión**: a partir de la secuencia “*seed*” encontradas en secuencia objetivo, el algoritmo extiende el alineamiento en ambas direcciones y continúa solo si la puntuación está por encima de un valor límite preestablecido.
- **Evaluación**: proceso mediante el cual el algoritmo filtra aquellos emparejamientos con secuencias repetitivas y evalúa el puntaje de los restantes teniendo en cuenta parámetros estadísticos.

PlasmiSPAdes

Software libre que a partir de los *reads* obtenidos en la WGS, ensambla y agrupa las secuencias correspondientes plásmidos. Esta herramienta implementa como método de clasificación la cobertura de *k-mer*, variable relacionada con la profundidad del ensamble. Ya que los plásmidos se replican en el cuerpo celular, es común que al momento de secuenciar el ADN bacteriano, los *reads* provenientes de plásmidos estén en mayor número de copias que los provenientes de la secuencia del cromosoma. Esta esta variación en las proporciones de los *reads* obtenidos, hace que se generen diferencias en

la cantidad de conexiones que se establecen en los gráficos de *bruijn* obtenidos por el programa, y que representan las relaciones entre los *reads* que provienen de plásmidos y los *reads* provenientes del cromosoma. la herramienta evalúa estas variaciones de conexiones para determinar que rutas o “edges” son de plásmidos (Antipov, Hartwick, et al., 2016a).

PPR-Meta

Diseñado para la identificación de fagos y plásmidos de forma unificada a partir de secuenciación de muestras de metagenómica, en fragmentos de secuencia ensamblados (*contigs*). Esta implementación de *deep learning* utiliza la metodología *Convolutional Neural Network*, la cual mejora la eficacia de la predicción en especial para la clasificación de los *contigs* más cortos (Fang et al., 2019). Ppr-Meta implementó para su entrenamiento un set de datos que consta de 10.090 genomas, 8.801 plásmidos y 2.279 fagos; en escenarios con los datos simulados, datos con 1% de las bases sustituidas y datos con 1% de las bases insertadas y eliminadas. Esto para evaluar la estabilidad de los resultados con errores que en la práctica pueden provenir de las metodologías de secuenciación.

AGMIAL

Pipeline libre, desarrollado en lenguaje java, con interfaz gráfica, que cuenta con múltiples herramientas bioinformáticas para la identificación y anotación de elementos genéticos en procariotas. Su método de clasificación es la Similitud de secuencias, sobre alineamientos entre el ADN problema y el ADN anotado de variadas bases de datos. Cuenta con dos subgrupos de herramientas llamadas PAM (*protein analysis manager*) (Wootton, 1994) y CAM (*contigs analisis manager)* (Lowe & Eddy, 1997), con la capacidad de ejecutar múltiples procesos bioinformáticos, secuenciales, y de modo automático (Bryson et al., 2006), permitiendo al usuario ocuparse de los conceptos importantes de la investigación y no de los procesos técnicos y repetitivos del procesamiento masivo de la información (Argemi et al., 2017).

BenchMarcking de herramientas de clasificación

Las herramientas *Plascope, PlasFlow y cBar* (Zhou & Xu, 2010) fueron comparadas utilizando un set de datos de 70 genomas de *Escherichia coli* que contenían plásmidos. En la Tabla 2, se agrupan los resultados en: *True positive, True negative, False positive y False negative*. A partir de estos valores se determinaron los parámetros de clasificación sobre los cuales se comparó la eficiencia de las herramientas: *Recall, Precision, Specificity y Accuracy*. Se evidenció el buen desempeño para clasificación de secuencias de plásmidos de la herramienta *Plascope* con las mejores métricas *de Recall, Precision, Specificity, Accuracy y F1 score*.

Tabla 2. Comparación herramientas de clasificación (Zhou & Xu, 2010)

	PlaScope	PlasFlow	cBar
True positive	1123	1106	954
True negative	9162	6231	5570
False positive	52	2983	3644
False negative	173	190	342
Recall	0.87	0.85	0.74
Precision	0.96	0.27	0.21
Specificity	0.99	0.68	0.6
Accuracy	0.98	0.7	0.62
F1 score	0.91	0.41	0.32

La Tabla 3, muestra la comparación del desempeño de las herramientas VirFinder (Ren, Ahlgren, Lu, Fuhrman, & Sun, 2017), VirSorter (Le Roux et al., 2015), PlasFlow y cBar; comparación en la que se tuvieron en cuenta los parámetros de calificación TPR(*Verdaderos positivos*), FPR(*Falsos positivos*) y AUC(*Area bajo la curva*). En dicha tabla se aprecia como *PlasFlow* tiene el mejor desempeño con relación a los verdaderos positivos, pero bajo en falsos positivos, lo que indica que sus restricciones no son lo suficientemente específicas y permiten un alto nivel de error. Por otro lado, PPR-Meta no es tan bueno en verdaderos positivos como PlasFlow, posiblemente debido a su diseño inicial abarca un mayor espectro de clasificación con fagos y plásmidos a diferencia de PlasFlow que es únicamente para plásmidos, mientras PPR-Meta en falsos positivos tiene

la mejor puntuación 14.14, mostrándose así como la herramienta con el mejor desempeño de acuerdo con el parámetro AUC.

Tabla 3. Evaluación de herramientas de clasificación de plásmidos

Tool	Evaluation on plasmid (%)		
	TPR	FPR	AUC
PPR-Meta	59.91	14.14	83.05
VirFinder	NA	NA	NA
VirSorter	NA	NA	NA
PlasFlow	71.89	62.59	56.30
cBar	52.68	46.07	53.31

La implementación de varias herramientas de clasificación de secuencias de plásmidos en un pipeline puede hacer una conjunción sinérgica de los atributos de cada metodología, es decir involucrar en una misma herramienta varias metodologías de clasificación que discriminan sobre distintas características biológicas de los plásmidos utilizando diferentes algoritmos Tabla 4.

Tabla 4. Algoritmos implementados

Algoritmos	Programas
Alineamiento	Agmial
Alineamiento Simth Waterman	Blast
Alineamiento y Indexación	Plascope
Bruijin graphs	PlasmidSPAdes
IA Redes profundas K-mer	PlasFlow
IA Redes profundas One-Hot Encode	PPR-Meta

Estas herramientas abordan el problema de clasificación de secuencias de plásmidos a partir de la secuencia del genoma completo, identifican en las secuencias varias características que son propias de los plásmidos. De manera concreta en las herramientas consultadas en el presente trabajo Tabla 5, se evidencia la recurrencia de la utilización de

Similitud de secuencias, uno de los métodos más confiables para identificar secuencias con un alto nivel de semejanza con secuencias de plásmidos que ya hayan sido reportadas, con base en métricas que determinan la similitud e identidad entre las secuencias problema y las conocidas. Su limitante está en que solo sirve para identificar secuencias de plásmidos previamente secuenciadas, anotadas que reposan en la base de datos de referencia.

Tabla 5. Principios de clasificación de plásmidos

Principio de clasificación	Programas
homología de secuencias	Plascope,Agmial,Blast
No de copias	PlasmidSPAdes
Reconocimiento de patrones	PlasFlow,PPR-Meta

Multi-clasificador

En el problema de la clasificación es de utilidad implementar varios clasificadores, “clasificadores débiles”, que brindan según su metodología especifica de decisión una propuesta de clasificación. Las características relevantes para el diseño de un multi-clasificador y su algoritmo de unificación, son el propósito de la presente sección.

El objetivo principal de diseñar e implementar un multi-clasificador es mejorar el desempeño de clasificación global, respecto al mejor de sus clasificadores débiles. Es decir, el multi-clasificador según sus métricas de clasificación se desempeña mejor que cualquier clasificador débil que lo compone, desempeño que depende de una característica importante llamada “diversidad”, propia de un algoritmo multi-clasificador y representa cuan variadas son las respuestas de los clasificadores débiles que le componen. El cálculo y comparación de la diversidad entre varios conjuntos de clasificadores débiles permite establecer que clasificadores débiles se deben incluir en el multi-clasificador, pues el costo computacional de estos algoritmos escala según la cantidad de clasificadores débiles que lo componen y es de utilidad excluir aquellos clasificadores que no hacen un aporte global de desempeño al modelo.

El cálculo de diversidad de un multi-clasificador puede realizarse de varias maneras ya que existen relaciones de diversidad pareada, es decir medir la diversidad entre 2 clasificadores débiles, o si se busca calcular la diversidad de un multi-clasificador diseñado con más de dos clasificadores hay métricas como Entropía, Varianza de Kohavi-Wolpert, medida de desacuerdo entre expertos, o medida de dificultad; las cuales se explican al detalle en (Cabrera-Hernández, Morales-Hernández, & Casas-Cardoso, 2016).

Una vez se ha decidido cuales clasificadores débiles se deberían incluir en el multi-clasificador, es necesario determinar el algoritmo de unificación, dentro de estos se encuentran:

Bagging* y *Hard Voting

Los algoritmos *Bagging* y *Hard Voting* son equivalentes pues utilizan las salidas de los clasificadores débiles como entrada, estos valores son normalizados para luego ser promediados en el caso de *Bagging*, y el algoritmo *Hard Voting* donde a todos los clasificadores se les atribuye el mismo peso en una sumatoria. Para estos dos algoritmos solo se configura un parámetro, el límite o frontera de decisión. Esta estructura de multi-clasificador donde la información se toma en paralelo, hace un cálculo directo, es decir este algoritmo no requiere de un entrenamiento, pero si de una optimización del parámetro limite o convencionalmente llamado *cut-off*. La Figura 3, muestra el esquema de funcionamiento del multi-clasificador con método *Hard Voting* (a) y *Bagging* (b), iniciando con los datos que cada uno de los modelos debe clasificar, representados con un circulo a la izquierda de los diagrama, luego cada clasificador débil genera un resultado de clasificación, estas salidas se normalizan y promedian para el caso de *Bagging* y únicamente se suman para el caso de *Hard Voting*, Entonces la decisión de los multi-clasificadores depende del límite de filtrado establecido por un proceso básico de optimización ((sanchezpares, 2019).

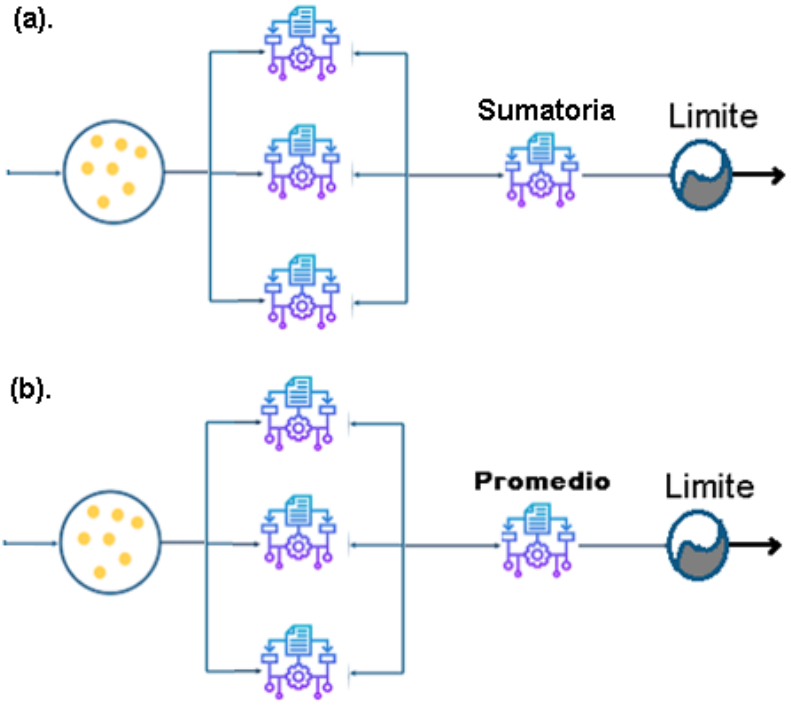


Figura 3. Funcionamiento Multi-clasificadores (a). *Hard Voting*, (b) *Bagging*.

Soft Voting

Es un algoritmo de cálculo directo que, como *Bagging* o *Hard Voting* no requiere de un entrenamiento, ya que implementa un coeficiente fijo para cada clasificador débil, es decir le da un mayor peso al clasificador débil que mejor se desempeña según la experiencia estadística. A diferencia de los dos métodos mencionados, *Soft Voting* requiere de una evaluación inicial del desempeño de los clasificadores débiles para ponderar su peso en la decisión final. Además, es necesaria una optimización para configurar el límite de decisión, como el descrito en la sección anterior Figura 4.

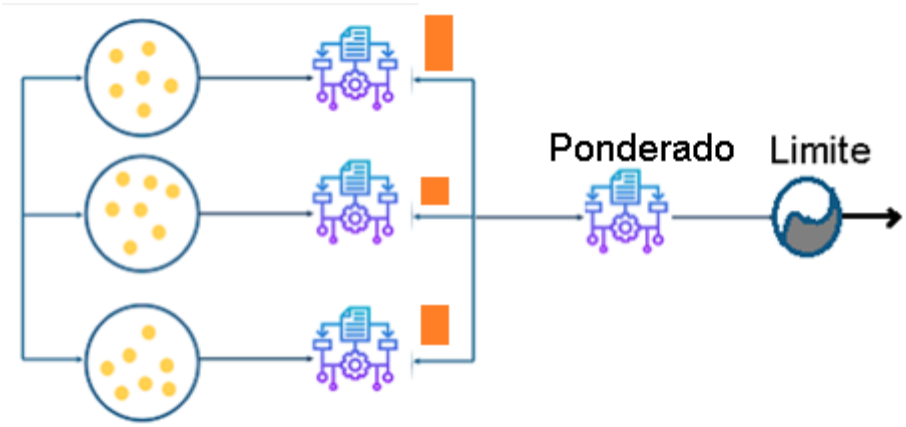


Figura 4. Estructura *Soft Voting* (sanchezpares, 2019)

Boosting

Algoritmo a diferencia de los multi-clasificadores mencionados, este determina los pesos de cada modelo según un proceso de entrenamiento, este proceso se hace en serie, ver Figura 5 (a) , buscando minimizar el error de cada herramienta, compensando el peso de la herramienta contigua en el flujo de entrenamiento, este algoritmo calcula un peso por clasificador débil, en la literatura llamado *alpha y* puede ser determinado de varias formas, ya que es un algoritmo con varias modificaciones y optimizaciones, el algoritmo clásico para el entrenamiento de este tipo de multi-clasificadores es **AdaBoost** Figura 6.

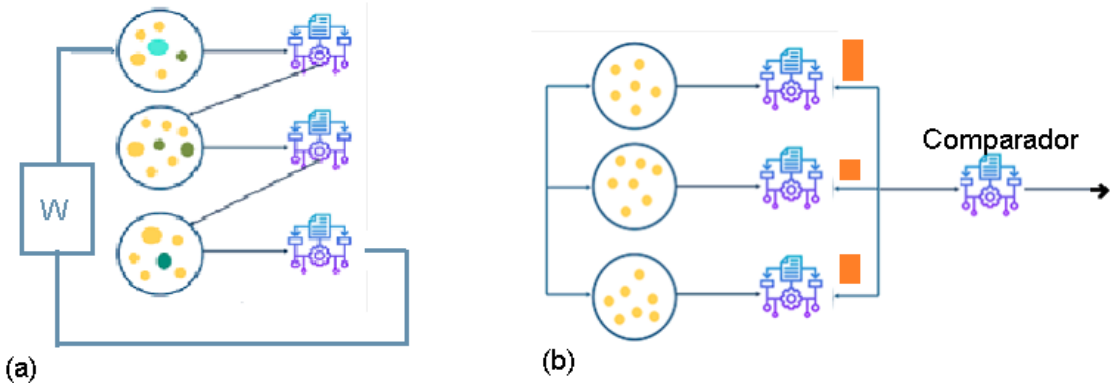


Figura 5. *Boosting* (a) Entrenamiento, (b) Implementación.

Luego del proceso de entrenamiento descrito, donde se buscan los *alphas* que generaron los valores mínimos de error durante el entrenamiento, la decisión de unificación del multi-

clasificador se hace comparando la sumatoria de los clasificadores débiles que comparten su decisión y tomando la clase que tenga una mayor votación Figura 5(b) .

Algorithm 1 *AdaBoost (Freund & Schapire 1997)*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n.$
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
   (c) *Compute*
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}.$$
   (d) *Set*
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, 2, \ldots, n.$$
   (e) *Re-normalize* w_i.
3. *Output*
   $$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

Figura 6. Algoritmo de AdaBoost.

Staking

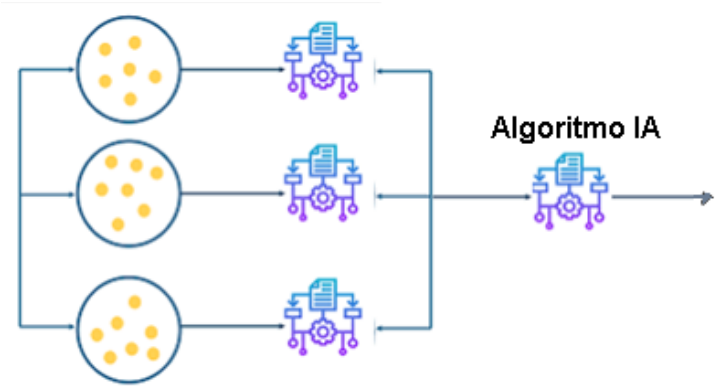


Figura 7. Estructura *Staking*

Es un algoritmo de unificación que hace un entrenamiento de un modelo de clasificación de inteligencia artificial que toma como características, las salidas binarias de cada clasificador débil Figura 7. La ventaja de esta metodología es que el modelo aprende desde

un set de datos como el mostrado en la Tabla 6, donde se pueden ver las columnas “Model #” con las decisiones de los clasificadores débiles y la columna “Actual_Output” la cual es usada en el entrenamiento para que el modelo aprenda de las decisiones singulares de cada clasificador débil. Este algoritmo al implementar inteligencia artificial permite generar fronteras de decisión más complejas que las posibles generadas con los multi-clasificador*es* clásicos presentados, ya que durante el entrenamiento encuentra relaciones entre las variables independientes, que un sistema determinístico no puede detectar.

Tabla 6. Set de datos de entrenamiento *Staking*

	Model-1 output	Model-2 output	.....	Model-n output	Actual Output
Training example -1	1	0		1	1
Training example -2	0	0		0	0
. . .					
Training example -3	0	1		0	0

Estos algoritmos de inteligencia artificial pueden estar basados en estadística como Descenso de Gradientes Estocástico (SGD) o en transformaciones lineales a un espacio característica sobre las variables independientes como lo hace *Suport Vector Machine* (SVM), en el entrenamiento, para la generación de funciones de mayor orden, es decir más complejas que permitan hacer una clasificación más apropiada sobre el espacio de búsqueda, o también implementar algoritmos evolutivos como es el caso de la colonia de hormigas en el diseño de un multi-clasificador (C., N., S., C., & G., 2017)

Justificación

Dada la exploración documentada en la anterior sección, se encontraron herramientas particulares que abordan el problema de la clasificación de plásmidos a partir de la secuencia del genoma completo con variadas metodologías y desempeños de clasificación. También se evidenció la ausencia de una herramienta que involucre distintas metodologías y unifique sus resultados. Por esta razón la implementación de distintas características (Similitud, *k-mer*, Coverage) de las secuencias plasmídicas para su identificación y posterior clasificación en un multi-clasificador pueden generar una

respuesta unificada robusta y de mayor calidad relativa en clasificación, que cualquiera de las metodologías que lo componen.

Pregunta de Investigación

¿De qué manera influye integrar los resultados de varias metodologías de identificación de secuencias plasmídicas a partir de secuencias obtenidas por secuenciación de genoma completo (WGS) de aislamientos de Klebsiella pneumoniae, en función del Recall, Precision, Specificity y Accuracy?

Objetivos

OBJETIVO GENERAL

Analizar el efecto de unificar 4 metodologías de clasificación de plásmidos existentes en una herramienta pipeline, para secuencias de genoma completo del microorganismo *Klebsiella pneumoniae,* con base en métricas de desempeño en clasificación.

Objetivos Específicos

- Generar un set de datos de prueba de genomas de *Klebsiella pneumoniae* que permita valorar el desempeño de cada herramienta bioinformática.
- Configurar cada módulo de clasificación específicamente para la sepa de trabajo en función del desempeño.
- Determinar el modelo de unificación de resultados del pipeline valorando el desempeño de cada uno de los módulos.
- Analizar el desempeño de la herramienta pipeline con los módulos de clasificación configurados, implementando el set de datos de *Klebsiella pneumoniae*.

2. Metodología

En el presente capítulo se presenta el proceso realizado para el desarrollo y optimización del pipeline**¡Error! No se encuentra el origen de la referencia.**, cuyo propósito es la identificación de secuencias asociadas a plásmidos a partir de lecturas, producto de la secuenciación de genoma completo por tecnología Illumina, dedicando cada sección del presente trabajo a: selección del set de datos de prueba, selección y puesta en marcha de las herramientas de clasificación, evaluación de las herramientas Spades, PlasmidSpades, selección por profundidad del Ensamble (script en python), Similitud de plásmidos y Similitud de cromosomas (script en python), evaluación de la herramienta PlasFlow, ya que este sistema de inteligencia artificial no tiene parámetros de configuración, pues su desempeño depende únicamente de la estructura de la red neuronal profunda y los datos usados en su entrenamiento por los diseñadores (P S Krawczyk, Lipinski, & Dziembowski, 2018a).

Selección y puesta en marcha de las herramientas de clasificación

Para el desarrollo del presente pipeline, como se planteó en el anterior capítulo se realizará un ensamble de lecturas con la herramienta Spades cuyo propósito y algoritmo se ha desempeñado mejor que Velveth en ensambles de genomas bacterianos (*Benchmarking of de novo assembly tools: SPAdes 3.9 vs Velvet 1.2*, n.d.), además una estrategia para la identificación de plásmidos desde *contigs* del genoma completo que implementa:

- La Similitud de secuencias de cromosoma y de plásmidos con un script en python3.7 de implementación de Blast (Program & Guide, 2009).
- Ensamble de *contigs* de plásmidos usando PlasmidSpades (Antipov, Hartwick, et al., 2016a).
- Discriminación de *contigs* por profundidad con la implementación de un script en python3.7.
- Discriminación de *contigs* de plásmidos por Inteligencia Artificial utilizando PlasFlow (P S Krawczyk, Lipinski, & Dziembowski, 2018a).

Herramientas sobre las cuales se abordarán sus configuraciones y características en el proceso experimental detallado en la sección de Resultados y Análisis.

Set de datos

Para determinar objetivamente el desempeño del Pipeline, se seleccionó un set de datos, del cual se conoce para cada aislamiento: las secuencias completas tanto del cromosoma como plásmidos, para determinar cuantitativamente la efectividad de cada módulo de clasificación del algoritmo. Se tomaron secuencias de *Klebsiella pneumoniae* de la base de datos *Genome* del NCBI que cumplan con un nivel del Ensamble descrito como completo, con *reads* almacenados en la base de datos SRA (Sequence Read Archive), obtenidos por tecnología Ilumina Miseq Figura 8.

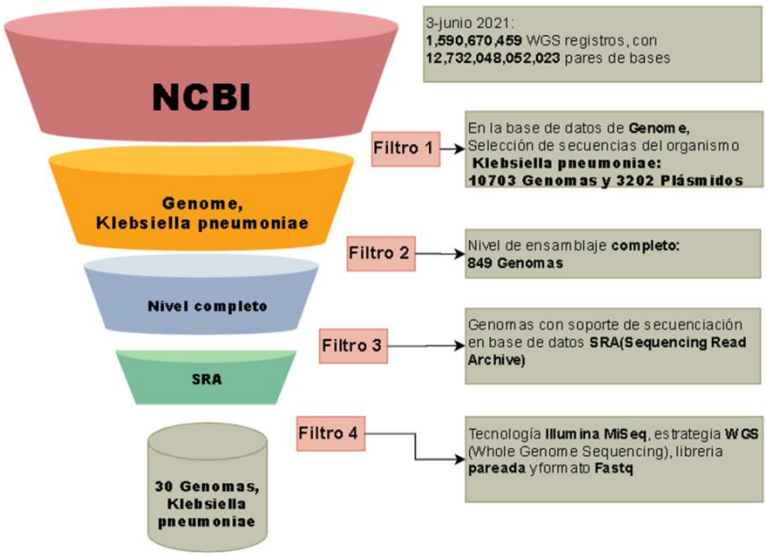


Figura 8. Generación, Set de datos

Pre-Procesamientos de datos

A las lecturas de secuenciación obtenidos del SRA, se les realizó una evaluación general de sus características con la herramienta FastQC v0.11.5, para conocer la calidad de las secuencias que serán la base de los siguientes procesos y análisis bioinformáticos.

De acuerdo con el resultado de calidad obtenido con FastQC para cada conjunto de lecturas, se realizó un proceso de limpieza con la herramienta Trimmomatic v 0.36, con la que se buscó eliminar segmentos de secuencia y secuencias de baja calidad, y secuencias correspondientes a los adaptadores de utilizados en los procesos de elaboración de las librerías de secuenciación. Después del proceso de limpieza de los *reads* se hizo de nuevo una revisión de calidad con la herramienta FastQC.

Ensamble de *novo*

Después del pre-procesamiento los *reads* fueron ensamblados con Spades v3.15.0. Los parámetros seleccionados en su ejecución fueron:

- --careful: Función que permite al programa consumir más recursos, a cambio de subprogramas de corrección de errores.
- -m (70): Función para limitar el consumo máximo de memoria.
- -t (20): Definición de la cantidad de hilos a usar en los procesos, si están disponibles.
- (-1,-2,--s1,--s2): Argumentos especificados para tomar los archivos pareados y no pareados producto de la limpieza de *reads.*
- -k(k1,,,, kn):La elección de los *K-mers* necesarios para el Ensamble se buscaron mediante el procedimiento descrito a continuación.

Elección de los tamaños de *k-mer* para Ensamble de novo, Herramienta Spades

Para la selección de los conjuntos de tamaños de *k-mer* se utilizó un script en Python 3.7 que crea combinaciones aleatorias de dichos conjuntos, usando la librería de NumPy, ensambla con Spades v3.15.0 los *reads* con cada conjunto de *k-mer*s, y genera un reporte con las métricas de cada ensamble (N50, Tamaño Genoma, *Contig* más largo, cantidad de *Contigs*) en los directorios de cada muestra, usando QUAST v5.0.2. El Ensamble final de los conjuntos de *reads*, se realizó con el conjunto de *k-mer*s con mejores resultados.

Evaluación y optimización de clasificadores débiles

El presente capitulo dispone una sección específica para cada clasificador débil, donde se detalla el procedimiento bioinformático realizado con el set de datos, para la exploración,

evaluación, modelado y optimización de los parámetros de configuración, optimización realizada para maximizar las métricas de desempeño en clasificación detalladas en función de (verdaderos positivos TP, falsos positivos FP, verdaderos negativos TN y falsos negativos FN) ver Ecuación 1. También se detalla los procesos y recursos físicos y de software para la implementación de estos experimentos.

Ecuación 1. Métricas de clasificación.

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1\,Score = \frac{2 * (Recall * Precision)}{(Recall + Precision)}$$

Clasificación de plásmidos con PlasmidSpades.

Los tamaños de *k-mer* utilizados para los ensambles *de novo* se usaron también para el ensamble con PlasmidSpades v3.15.0. Dado que las demás herramientas utilizadas para la clasificación de secuencias plasmídicas usan los *contigs* obtenidos con Spades, fue necesario determinar qué información genética de los *contig|s* producidos por PlasmidSpades, son los correspondientes a los *contigs* producidos por Spades, para utilizar estos últimos en la etapa final de clasificación de las secuencias plasmídicas.

Los parámetros seleccionados en su ejecución fueron:

- --plasmid: Función para indicar el ensamble únicamente de secuencias plásmidicas.
- --careful: Función que permite al programa consumir más recursos, a cambio de subprogramas de corrección de errores.

- -m (70): Función para limitar el consumo máximo de memoria.
- -t (20): Definición de la cantidad de hilos a usar en los procesos, si están disponibles.
- (-1,-2,--s1,--s2): Argumentos especificados para tomar los archivos pareados y no pareados producto de la limpieza de *reads.*
- -k(k1,,,,kn):La elección de los *K-mers* necesarios para el Ensamble se buscaron mediante el procedimiento descrito a continuación.

Representación de PlasmidSpades en *Contigs* Spades

Para definir que *contigs* de Spades están representados en las secuencias ensambladas por PlasmidSpades fue necesario ejecutar un alineamiento con Blast v 2.10.1, usando las siguientes instrucciones y argumentos:

- makeblastdb: para la definición y preparación de información genética como base de datos, usando los siguientes parámetros:
  - -in: *contigs* de Spades como base de datos.
  - -dbtype 'nucl': especificando el formato de la información genética.
  - –out: especificación, ubicación de los resultados.
- Blastn: Función de ejecución del alineamiento local entre *contigs* de Spades vs PlasmidSpades, usando siguientes los parámetros:
  - –*query*: secuencias a identificar, en este caso los componentes de PlasmidSpades
  - –db:Uso de la base de datos generada en la anterior función “makeblastdb”
  - -max_target_seqs(1): Límite superior de cantidad de alineamientos permitidos por target.
  - -max_hsps(3): Cantidad de alineamientos incluidos en el archivo salida por cada *query*
  - –task(blastn): Tipo de alineamiento de las secuencias, en este caso nucleótido vs nucleótido.
  - –dust(no): No incluir un enmascarado de zonas repetitivas.
  - -outfmt ("10 qseqid sseqid pident qcovus length slen qlen qstart qend sstart send"): Especificación del formato de salida.
  - –out: especificación de destino de archivos generados en la ejecución.

En función de las métricas de los alineamientos: Porcentaje de Identidad, Cobertura de la secuencia *Query* y Cobertura de la secuencia Subjec (%ID, %qCov y %Scov respectivamente), se buscó, a través de un experimento iterativo de dos variables independientes; porcentaje de identidad y porcentaje de cobertura(*Query* y Subject), aquellos límites de decisión, que maximizan una elección correcta de aquellos componentes de PlasmidSpades, representados en Spades, a través del diseño de un script en Python 3.7, que para cada muestra del set, lee la tabla producto del alineamiento Blastn entre Spades y PlasmidSpades y selecciona los *contigs* que pasen el filtro de %ID, %qCov y %Scov. La manera de calificar los variados límites de decisión se hizo comparando aquellos nodos que pasan los filtros contra los identificados manualmente en Mauve v 20150226, para de esta manera calcular las métricas Precisión, *Recall* y *F-Score*.

Clasificación por profundidad de Nucleótido relativa.

Al ejecutar Spades sobre los *reads* limpios, se obtiene un archivo multi-fasta con los *contigs* ensamblados, en donde cada uno de estos es identificado por Spades como:

>NODE_##_length_1682280_cov_162.837

- Node: Identificador entero del *contig* generado, el primer nodo es también el más largo del proceso de ensamble.
- Length: Numero entero que informa la cantidad de bases que tiene la secuencia contigua.
- Cov: Numero Flotante con la profundidad de *k-mer* promedio a lo largo del *contig.*

Ecuación 2. Calculo *Outlayer*s, Atípico y Extremadamente Atípico

$$VA = Q3 + 1{,}5 * RIC$$

$$VEA = Q3 + 3 * RIC$$

RIC= rango intercuartil, Q3-Q1

Para hallar el límite de profundidad al cual se clasifica un *contig* como posiblemente parte de un plásmido, se creó un script en Python *3.7*. Se planteó un experimento con un espacio de búsqueda basado en la distribución de los valores de profundidad promedio de los *contigs* de cada muestra, a partir de los cuales se calcularon tanto el límite de los valores atípicos (VA) como el de los valores extremadamente atípicos (VEA) (Ecuación 2). Para esto se calcularon el primer y tercer cuartil de toda la distribución de profundidad de todos los *contigs*. El experimento consistió en determinar, según la profundidad de cada *contig,* si esta era mayor o menor que el valor límite tomado dentro de los 100 puntos en que se dividió el intervalo entre él VA y el VEA (Figura 9), para determinar el límite de profundidad por encima de cual se maximiza la eficiencia de clasificación de *contigs* plasmídicos sobre los cromosomales.

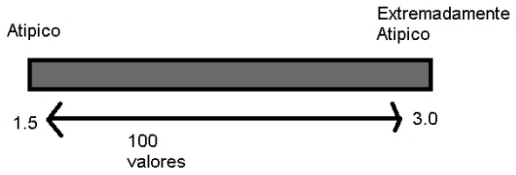


Figura 9. Espacio de búsqueda, Modelo profundidad de Nucleótido.

Clasificación por Similitud de secuencias, con Blastn.

En el diseño del módulo de clasificación de plásmidos a partir de secuencias homologas de una base de datos local, fue necesario crear un script en Python *3.7*, que permitiera de manera sencilla interactuar con ficheros y servicios de consumo web, para crear y actualizar un sistema de archivos de bases de datos de plásmidos y de cromosomas de bacterias específicas.

Las secuencias necesarias para consolidar las bases de datos, contra las cuales los métodos del Pipeline de clasificación por Similitud de Plásmidos y de Cromosomas, fueron obtenidas de bases de datos públicas (nuccore y genome) del NCBI a través de las librerías "request" y "xlml" de Python *3.7*, junto con las librerías necesarias para la administración de archivos y scripts externos llamadas "sys", "datetime","getopt", "datetime" y "email" librería útil para hacer una notificación al usuario por medio de correo electrónico.

Por otra parte, en la optimización, se buscaron aquellos parámetros que mejoren la eficacia de clasificación en la selección de secuencias similares contra base de datos de plásmidos y No similares contra base de datos de Cromosomas.

Para seleccionar aquellos secuencias similares y No-similares, se requiere establecer los límites de decisión sobre las métricas de los alineamientos: porcentaje de Identidad y porcentaje de cobertura del "*Query*", calculadas por Blastn en los alineamientos entre el genoma ensamblado por Spades y las bases de datos de plásmidos y de cromosomas. Se planteó un experimento para las dos aproximaciones, consistente en generar una calificación de *F-Score* para cada configuración de parámetros, como se indica a continuación:

- DB Plásmidos: porcentaje de identidad del alineamiento entre 80 y 100, porcentaje de cobertura del "*query*" por el "Subject" entre 20 y 100; resultando en 1600 configuraciones de parámetros distintas para cada muestra.
- DB Cromosomas: porcentaje de identidad del alineamiento entre 80 y 100, porcentaje de cobertura del "*query*" por el "Subject" entre 20 y 100; resultando en 1600 configuraciones de parámetros distintas para cada muestra.

Para realizar esto se implementó un script en Python*3.7* (AnexoG), en la plataforma Google Colaboratory, que califica la eficiencia de la clasificación comparando la respuesta de determinada combinación de %ID y %COV, contra las secuencias de evaluación, con ayuda de la librería Sklearn y el paquete Confusion_Matrix para obtener métricas de evaluación *recall, precisión y F-Score*.

Evaluación Herramienta PlasFlow

Como se mencionó al inicio del capítulo la herramienta PlasFlow v.1.1 no tiene parámetros de configuración para optimizar, pues esta herramienta tiene configurada su red neuronal con unos pesos definidos en el proceso de entrenamiento con una gran cantidad de secuencias de plásmidos. Los parámetros necesarios para su ejecución fueron:

- --input: Archivo fasta con los *contigs* problema.
- --output: Especificación del destino de los archivos resultado, generados.

Para la incorporación de los resultados de PlasFlow v.1.1 en el multi-clasificador, propósito de este trabajo, fue necesario tener una métrica objetiva de su desempeño en la clasificación, para la cual se diseñó de un script en Python3.7 que se utilizó en la plataforma Google Colaboratory, que compara los *contigs* que Plasflow clasifica como plásmidos contra los *contigs* verdaderamente pertenecientes a plásmidos, identificados previamente a través de un proceso manual a partir del alineamiento, usando Mauve, de los *contigs* obtenidos por Spades y los plásmidos descargados de la base de datos Genome del NCBI (set de *contigs* verdaderos), generando una matriz de confusión que permite calcular los valores de desempeño *F-Score*, *recall* y *precision*.

Multi-Clasificador

El algoritmo de unificación de resultados se diseñó con el propósito de generar un consenso de los *contigs* pertenecientes a plásmidos con base en los resultados de los

clasificadores débiles: PlasmidSpades, PlasFlow, Profundidad, Similitud plásmidos y No-Similares cromosoma.

Para diseñar este algoritmo fue necesario resolver dos preguntas:

- ¿Cuál es la mejor metodología de unificación de resultados a implementar?
- ¿Cuál es el conjunto de clasificadores débiles que deben componer el multi-clasificador?

Para resolver estas preguntas, en primer lugar utilizando un script en Python 3.5 que utiliza las librerías Os, Bio, el método Python open() y *Numpy* (Anexo I), se generó una tabla con todos los genomas del set de datos, en las filas están los nombres de los *contigs* etiquetados por Spades, y en las columnas, el resultado de cada clasificador débil con respecto a cada *contig*, de tal forma que si el clasificador débil lo selecciona como plasmídico el valor en la columna será '1' y de lo contrario será '0', si el clasificador lo identifica como cromosoma. También se agregó una columna con la identificación real de cada *contig*, basada en el set de *contigs* verdaderos.

Para dar respuesta a las preguntas planteadas, con base a lo reportado en la literatura, se implementó y ejecuto en Google Colaboratory un script en Python V*3.7* (Anexo H), que permitió implementar los diferentes métodos de unificación seleccionados, con diferentes parámetros específicos para cada Multi-Clasificador y diferentes conjuntos de clasificadores débiles, buscando los valores con mejor desempeño, en base a los mismos subconjuntos del se datos (entrenamiento/evaluación), para hacer una comparación relativa entre las herramientas.

Los experimentos de optimización sobre los métodos seleccionados son los siguientes:

- *Bagging*: En esta implementación para cada una de las 26 posibles combinaciones de 2, 3, 4 y 5 clasificadores débiles, se probaron distintos límites de decisión en el rango de 0 a 1 en intervalos de 0.01. Se determinaron los valores de F score para cada una de las 2600 pruebas para medir el desempeño en clasificación y extraer el máximo.

- *Hard vote:* En esta implementación para los 26 posibles conjuntos de clasificadores débiles, se probaron límites de decisión en el rango 1 a 4, con variaciones de una unidad +-1. Se determinaron los valores de F score para cada una de las 77 pruebas para medir el desempeño en clasificación y extraer el máximo.
- *Boosting*: Para este Multi-clasificador que implementa el algoritmo de entrenamiento descrito en el capítulo 1, algoritmo de entrenamiento iterativo, que busca aquellos pesos, que multiplicando el resultado de los clasificadores débiles, minimiza o compensa el error de la herramienta previa, este proceso consta de: Generar 26 conjuntos de clasificadores débiles, grupos entre (2,3,4,5) clasificadores, con sus respectivas salidas y agregando la columna con la predicción del algoritmo multi-clasificador “df_boosting”. Luego cada conjunto es entrenado iterando el set de datos 5 veces, este proceso permite obtener un peso para cada clasificador. Con los pesos de cada clasificador definido, se compara el desempeño de estos multi-clasificadores, e identifica el máximo desempeño de todos los conjuntos de clasificadores débiles, evaluados con los pesos encontrados en el proceso de entrenamiento, en total 26 valores fueron comparados.
- *Staking*(SVM): Multi-clasificador que implementa Support Vector Machine como modelo de Inteligencia Artificial, para su entrenamiento se implementa el siguiente proceso: Generar los 26 conjuntos de clasificadores débiles entre (2,3,4,5), genera un *dataframe* con sus respectivas entradas y salidas, con la implementación del método “train_test_split” de la librería Sklearn para cada *dataframe,* se dividen los conjuntos de entrenamiento y evaluación, se entrena el modelo con el método “fit” de Sklearn, usando como argumento la matriz de entrenamiento, usando las clasificaciones como características y la identidad real de los *contigs* como referencia, Al final se comparan los valores *f-score*, con los métodos “confusión_matrix” y “classification_report”, del modelo entrenado, usando el conjunto de evaluación.
- *Soft Voting*: Este método no requiere de un entrenamiento, pero sí de una evaluación objetiva sobre cada clasificador débil, pues el resultado de esta metodología de clasificación es una suma ponderada de los resultados de clasificación por un peso relativo al comportamiento de cada clasificador. Además, como en los métodos *Bagging* y *Hard vote*, es necesario definir un límite de decisión entre 0 y 1 óptima.

Este límite se calculó de la siguiente forma: Sobre los 26 posibles conjuntos de clasificadores débiles, se generó un *dataframe* con sus respectivas entradas y salidas, luego para todo el set de datos, se generó una evaluación del desempeño, una para cada límite de decisión entre 0 y 1, con una resolución de 0.01, es decir 100 evaluaciones, por último, se compararon los resultados de todos los conjuntos contra todos los límites de decisión, es decir 2600 valores de F-score, para determinar el conjunto y limite que maximiza la eficacia de las predicciones.

El mejor Multi-modelo, se identificó comparando el desempeño de los 5 valores máximos de los modelos de unificación (*Bagging*, *Hard vote, Boosting, Staking*(SVM) y *Soft Voting*), con su respectivo conjunto de clasificadores, optimizados anteriormente.

Pipeline

La implementación del script bioinformático de identificación y anotación de plásmidos y cromosomas, sobre lecturas de secuenciación NGS, se realizó como se indica a continuación:

Análisis de necesidades: basadas en las necesidades de configuración de las distintas herramientas, la obtención de datos a partir de bases de datos externas, la generación de bases de datos locales, la interoperabilidad de los distintos set de datos y funcionalidades de interacción con el usuario y verificación de errores.

Diseñó del algoritmo: diagrama de flujo que implementa los pasos y procesos definidos en el análisis de necesidades, notificaciones desde el servidor hacia el usuario y las posibilidades de falla, con las respectivas respuestas del sistema.

Implementación del pipeline: se realizó en un servidor Dell PowerEdge M640 con 92 Gb de RAM, doble procesador Intel Xeon de 16 cores, y sistema operativo OpenSUE Leap 15.3. Para la implementación del diagrama de flujo diseñando en la etapa anterior se usó Python 3.7 y las librerías: datetime, time, copy, time, sklearn.metrics, os, getopt,

csv, sys, copy, difflib, glob, math, pprint, sutil, re, Bio, numpy, statistics, typing, email.mime.text, smtplib, gzip, joblib, traceback.

Evaluación pipeline frente a secuencias de *Pseudomonas Aeruginosa.*

El funcionamiento de la herramienta propuesta en este trabajo, la cual fue entrenada con secuencias genéticas de *Klebsiella pneumoniae,* fue evaluado usando datos de *Pseudomonas aeruginosa*, para determinar si es generalizable a otras especies bacterianas.

Esta evaluación es descrita a continuación:

1. Set de datos: 15 genomas de *Pseudomonas Aeruginosa, obtenidos de las bases de datos Genome y SRA,* con los mismos criterios expuestos en la sección “Diseño Set de Prueba”.
2. Evaluación y Filtrado: El conjunto de *reads* provenientes del SRA es evaluado con Fastqc y filtrado por Trimomatic.
3. Ensamble *de novo:* los *reads* filtrados son ensamblados *de novo*, con la herramienta Spades usando los *k-mer* propuestos por la herramienta (21, 33, 55, 77, 99, 127).
4. Identificación de *Contigs* plasmídicos: con la herramienta Blastn se genera un alineamiento de los ensambles *de novo*, con las secuencias completas de Plásmidos y Cromosomas obtenidas de la base de datos Genome, clasificando como plásmidos aquellos *contigs* que alinean con métricas de porcentaje de identidad y cobertura del *query* confiables contra la base de datos de plásmidos y también aquellas secuencias que no alinean bien contra la base de datos de cromosoma.

5. Identificación de Plásmidos con el pipeline: usando el pipeline propuesto en este trabajo se ejecuta todo el proceso de identificación con las configuraciones y modelos encontrados en la metodología.
6. Evaluación: Se evaluó el desempeño del Multi-clasificador, comparando los *contigs* verdaderos y etiquetados, obtenidos en el paso 4, contra los reportados por el pipeline, para calcular las métricas *F-Score*, *recall* y precisión; esta evaluación se realizó con un script en Python 3.7, en el servidor, cuyo requerimiento especifico es la librería “joblib”, para poder usar el *multi-clasificador* de clasificación, entrenado en los servidores de Google (Anexo L).

3. Resultados y Análisis

Set de Datos

El Anexo A contiene los 30 genomas *Klebsiella pneumoniae* que cumplen las condiciones de: nivel del ensamble descrito como completo, *reads* almacenados en la base de datos SRA (Sequence Read Archive) y secuenciados por tecnología Ilumina Miseq. En la tabla se aprecia el código del cromosoma en el NCBI, la etiqueta para el experimento, tamaño del cromosoma, contenido de GC, código SRA de la secuenciación, código del experimento responsable, numero de plásmidos y etiquetas de cada plásmido. La selección de 30 como número de genomas del set de datos de prueba, se tomó con base en los utilizados en evaluaciones de herramientas similares como lo son PlasFlow v.1.1 (P S Krawczyk, Lipinski, & Dziembowski, 2018b), PlasmidFinder (Carattoli et al., 2014) y PlasmidSpades (Antipov, Hartwick, et al., 2016b) que utilizaron sets de prueba desde 6 a 49 genomas bacterianos.

Se seleccionó este set de datos con el objeto de enfrentar las herramientas de selección de secuencias plasmídicas con datos y variaciones biológicas y experimentales reales, ya que la alternativa de usar datos simulados, como se hace en muchos trabajos de este tipo, no considera completamente la variación biológica y experimental real. La descripción general del set de datos construido es:

- Tamaños de genoma entre 5.203.126 y 5.616.605 pb, un promedio de cromosomas de 5.401.074 pb, valor dentro del rango de los reportados en el NCBI.
- Variaciones en el contenido de plásmidos de 0 a 8, con el objeto de observar el desempeño de los clasificadores, ante genomas con variaciones en el numeró y tamaño de plásmidos.
- Porcentaje de GC entre 56.17 y 57.6, variación que permite evaluar el desempeño de herramientas como PlasFlow, cuyo algoritmo usa patrones de contenido k-mer, relacionados de manera general con el contenido de GC.

Limpieza de lecturas de secuenciación

El AnexoB muestra los resultados de la evaluación inicial en las columnas de la sección de preprocesamiento, en las cuales se evidencia que los *reads* de las muestras tenían una calidad variable, algunos de ellos almacenados en el NCBI sin ningún pretratamiento, como lo evidencia el resultado de calidad y el valor constante del tamaño de los *reads* de las muestras (color naranja AnexoB). Por otro lado, los sets de *reads* que tuvieron un filtrado de calidad previo, no contenían adaptadores de secuenciación y tienen un rango entre 31 y 301, en el tamaño de los *reads*. Efecto común después de una limpieza.

Tras esta primera evaluación se procedió al filtrado de todos sets de datos utilizando Trimmomatic v 0.36. Los *reads* obtenidos fueron evaluados con FASTQC nuevamente, mostrando que el proceso de filtrado elimino tanto las secuencias y segmentos de secuencia de baja calidad, como los adaptadores Illumina presentes. Como era de esperarse hubo una disminución en el número de *reads* de cada set de datos que estuvo en el rango de 0 a 58%, mostrando una calidad variable en los datos utilizados, que hacen mas cercanos a la realidad los resultados de la evaluación de los distintos métodos de selección de secuencias plasmídicas

Ensamble de *novo*

El experimento que evalúa todas las posibles combinaciones de tamaño 6, utilizando un diseño experimental (Otzen & Manterola, 2017), donde se hizo exploración aleatoria en el rango 21-127pb, evaluando el comportamiento de las métricas de ensamble (NG50, No_*Reads* y Errores). Debido a los requerimientos en términos de espacio y tiempo, se seleccionaron del set de datos de 30 genomas, 3 genomas: SRR9042857, SRR6514350 y Kp009. SRR9042857 sin presencia de plásmidos, SRR6514350 con la mayor cantidad de plásmidos (8 plásmidos) y *Klepsiella pneumoniae* Kp009 aislada y secuenciada en el Instituto de Biotecnología de la Universidad Nacional. El NG50 se obtuvo tomando un tamaño de genoma promedio de 5.7Mb.

Para esta primera parte se seleccionó al azar para cada aislamiento un total de 50 combinaciones de 6 tamaños de *k-mer*, en el rango 21 a 127. Estos 150 ensambles resultantes se evaluaron con la herramienta Quast y desde un script en Python v3.7 se

generó un archivo unificado con las métricas extraídas de cada ensamble, a partir de los archivos “report.txt” generados por Quast. Sobre este archivo unificado “tabla_ensamble.txt“, se hizo una evaluación grafica (Figura 10)**¡Error! No se encuentra el origen de la referencia.**, donde se observó el comportamiento de las métricas para la muestra SRR6514350 titulada “350”, la cual tiene el mismo comportamiento general que las otras dos muestras, como se aprecia en el AnexoC, y que permitió evaluar el efecto del tamaño de los *k-mers* sobre el NG50, el número de_*contigs* y la cantidad de errores. Se aprecia como las combinaciones de *k-mers* con mayores tamaños (90-127) maximizan el NG50 y minimizan la cantidad de *contigs*. Por otra parte, se evidencia que la cantidad de errores es poco sensibles a las variaciones de *k-mer* pues su variación oscila entre 2 y 7, siendo más baja cuando el tamaño de *k-mer* es bajo.

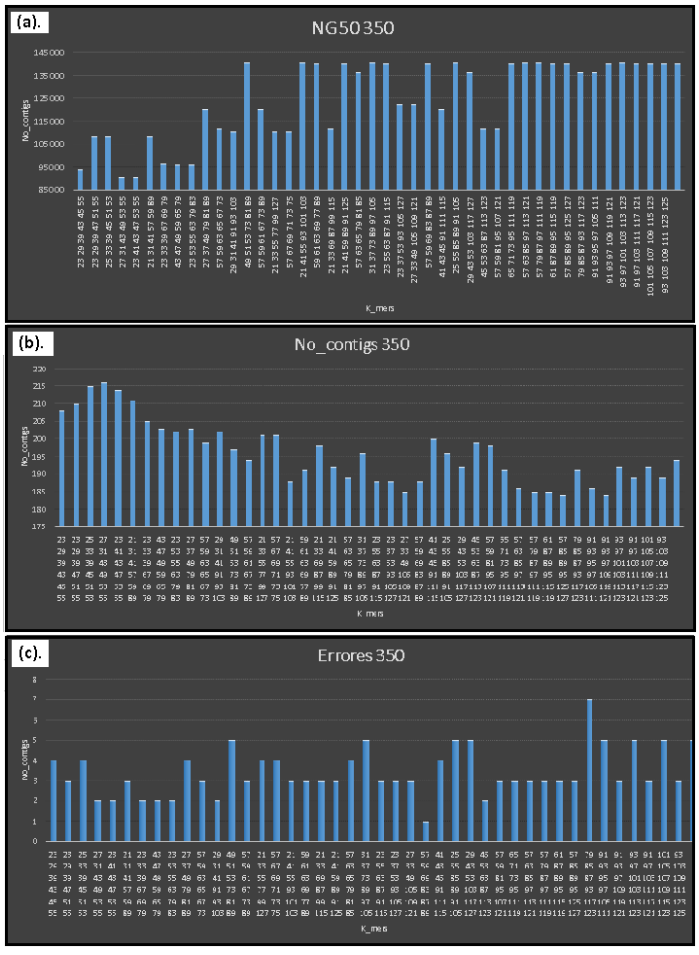


Figura 10. Evaluación Muestra "350" todo el espectro: Eje Y métrica de evaluación (a)NG50, (b)Numero de *Contigs* y (c) Numero de Errores, Eje X combinación de 6 *k-mer*.

Gracias a esta exploración del rango completo se hizo una búsqueda aleatoria simple sobre los tamaños grandes de *k-mer* (90-127 pb), pues allí se evidenció el mejor comportamiento de las métricas más relevantes NG50 y No_*contigs.* La cantidad de errores es casi invariable para todo el espectro ya que estos pueden ser efecto de la metodología, librería, calidad de secuenciación y las zonas repetitivas, más que un efecto directo de los tamaños de *k-mer* en el ensamble *de novo.*

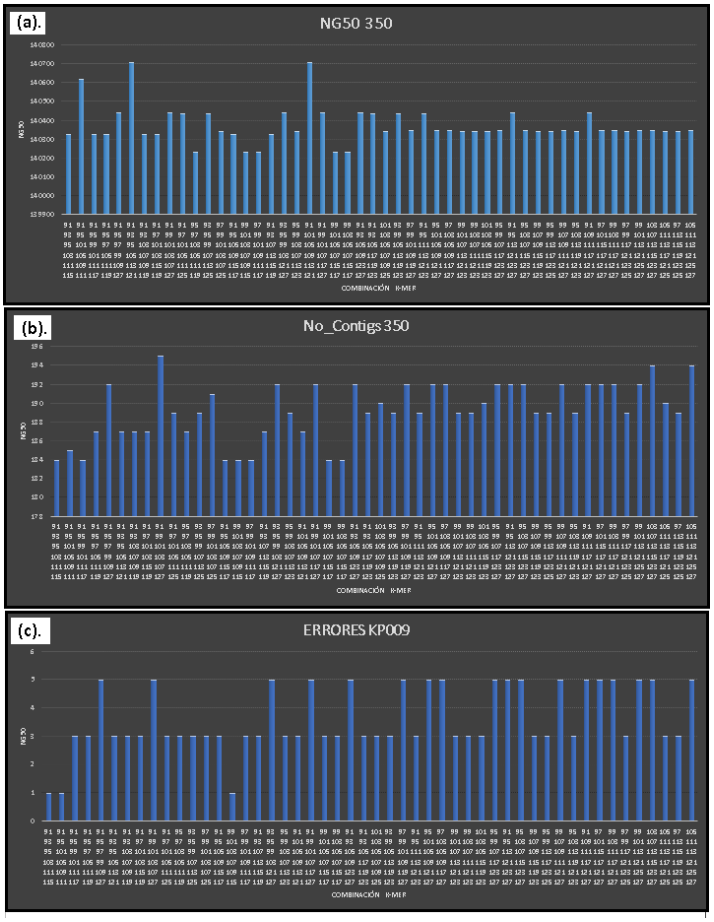


Figura 11. Evaluación Muestra "350" espectro grande: Eje Y métrica de evaluación (a)NG50, (b)Numero de *Contigs* y (c) Numero de Errores, Eje X combinación de 6 *k-mer* para cada ensamble.

Esta evaluación se implementó utilizando el script usado en la anterior exploración, reconfigurando el espacio de búsqueda de *kmers* entre 90 y 127 pb y generando 50 ensambles por muestra. El análisis de las métricas de ensamblaje se aprecia en Figura 11, donde para el NG50 no se puede observar una dependencia de los tamaños de *k-mers*, aunque algunas combinaciones puntuales tuvieron el mejor desempeño sin ninguna correlación aparente. Por otro lado, las métricas No_*Contigs* y Número de errores, muestran una ligera tendencia a tener buenos desempeños en ensamblajes obtenidos con valores bajos de tamaño de *kmer*, aunque de una manera no concluyente. Lo anterior indica que posiblemente habría que explorar todo el espacio de búsqueda de las combinaciones de *kmers* (~22 millones), para encontrar el máximo local, que posiblemente sería diferente para cada genoma estudiado, y difícil de calcular debido a la cantidad de tiempo y espacio de almacenamiento requeridos.

Para efectos prácticos de funcionamiento se hizo una comparación de ensambles de las 3 muestras seleccionadas para el ensayo de búsqueda aleatoria, usando Spades con la mejor combinación de *k-mer* resultado del experimento de búsqueda aleatoria, Spades usando la combinación de *k-mers* por *default* , Unicycler v0.4.8 y MegaHit v1.2.9 con y sin la función “mercy”, que según el autor permite generar un ensamble más completo debido a que incluye las lecturas con una baja profundidad al final del ensamble para rellenar huecos (Sadakane et al, 2015). Los resultados de esta comparación se aprecian en la **¡Error! No se encuentra el origen de la referencia.**, donde se pueden comparar los distintos desempeños en función de las siguientes métricas: No_*Contigs*, *Contig* mas largo, tamaño total de ensamble, NG50, NG75, LG50, LG75, cantidad de errores, porcentaje del tamaño del genoma cubierto, combinación de *k-mer*s implementados, el tiempo de ejecución y cantidad de hilos utilizados durante el proceso de computo.

Tabla 7. Comparación de herramientas de Ensamble. En naranja se etiquetan aquellas métricas con el mejor valor obtenido para cada muestra, en amarillo se etiquetan las

variaciones de desempeño y en blanco se etiquetan los desempeños sin ninguna variación.

muestra	Ensamble	No_contigs	C_Mas_Largo	Total_Ensamble	NG50	NG75	LG50	LG75	Errores	%_del_Genoma	K-mers	tiempo(min)	hilos
350	Mega-no-mercy	11114	316592	12725413	134131	75949	14	28	178	99,873	21-255	145	8
	Mega-mercy	11042	356176	12649730	110932	60819	15	30	173	99,872	21-255	144	8
	unicycler	219	315633	6081911	140149	59695	14	29	0	97,831	127	59	10
	Spades_def	197	316134	6145750	140440	60105	14	29	5	98,272	21-33-55-77-99-127	24	10
	Spades Exp	184	316114	6140948	140234	60244	14	29	3	98,276	97 99 101 109 111 117	22	8
857	Mega-no-mercy	249	792980	5312397	427024	248205	5	9	7	99,803	21-255	70	8
	Mega-mercy	249	792980	5312397	427024	248205	5	9	7	99,806	21-255	73	8
	unicycler	32	1085630	5171457	369886	212525	5	10	0	99,059	127	24	10
	Spades_def	31	1048876	5183987	370140	285953	5	9	1	99,265	21-33-55-77-99-127	20	10
	Spades Exp	28	1048874	5184037	381680	285951	5	9	1	99,272	101 111 113 115 121 125	16	8
kp009	Mega-no-mercy	398	625311	5682845	197785	100446	9	17	118	1,088239625	21-255	82	8
	Mega-mercy	391	621898	5684641	197785	100446	9	17	114	1,088583551	21-255	83	8
	unicycler	110	621262	5406102	197965	86314	9	18	48	1,035244568	127	63	10
	Spades_def	189	621888	5530183	224705	105999	9	16	50	1,059005529	21-33-55-77-99-127	23	10
	Spades Exp	174	621884	5502884	224701	94728	9	17	79	1,053777892	97 105 113 119 121 125	21	8

La herramienta que presentó el peor desempeño fue MegaHit, en cuanto a número de *contigs* y cantidad de errores (Tabla 7). Se pudo establecer que la combinación de *k-mer*s utilizada por Spades por *default* (21,33,55,77,99,127), logró en la mayoría de las métricas los mejores desempeños. Esto indica que una búsqueda exhaustiva de combinaciones de *k-mer* para un ensamble *de novo*, invirtiendo tiempo de máquina y capacidad de almacenamiento considerables, se desempeña casi de la misma manera que implementado los *k-mer* por *default* que recomienda Spades, por lo que se decidió hacer el ensamblaje de los genomas de *Klepsiella pneumoniae* con Spades utilizando su combinación de *k-mer* por default.

Evaluación y optimización de clasificadores débiles

A continuación se detallaran los procesos de evaluación y optimización de cada clasificador débil implementado en el Pipeline. El proceso general de configuración de parámetros aplicado a todos los modelos es descrito en la Figura 12**¡Error! No se encuentra el origen de la referencia.**, donde el set de datos en amarillo, corresponden a las secuencias plásmidos y cromosomas completos, y los respectivos *reads* obtenidos por metodología Illumina, descargados de las bases de datos Genome y SRA del NCBI, cumpliendo criterios detallados en la sección Diseño set de prueba.

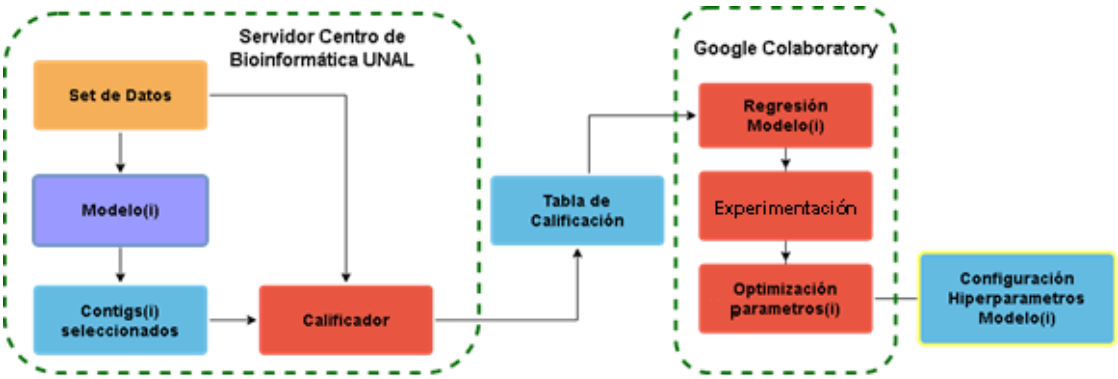


Figura 12 Proceso General de Optimización

Con estos datos se evaluó el desempeño de cada clasificador débil Modelo (i) (Figura 12), variando sus parámetros específicos dentro del espacio de su desempeño, para producir los *contigs*(i) seleccionados como plasmídicos. Los *Contigs* (i) seleccionados por cada modelo (i), fueron evaluados a través del script "Calificador" Figura 12, generando la "Tabla de Calificación" con los resultados de *precisión, recall* y *f-score*. Utilizando Google Colaboratory, con Python v3.7, se hizo un proceso de optimización de la selección de parámetros para cada Modelo (i), donde inicialmente se buscó una función "regresión", que representara el comportamiento de las métricas en función de los parámetros de configuración, y luego los (experimentos y optimización) (i), permiten buscar aquella configuración de parámetros que generan la mayor eficacia de clasificación en cada clasificador débil, estos procesos de optimización, se describen a detalle para cada clasificador débil en las siguientes secciones.

Clasificación de plásmidos con PlasmidSpades

PlasmidSpades, al igual que Spades, está basado en un algoritmo Bruijin de solución de grafos, y a diferencia de este último, incluye dos procesos posteriores: la evaluación de secuencias circulares propias de plásmidos y la profundidad relativa de las secuencias, esta última evaluada respecto al promedio del genoma, la cual generalmente es mayor en plásmidos. Este proceso genera *contigs* mucho más largos que los ensamblados por Spades.

La diferencia entre los *contigs* ensamblados PlasmidSpades y los generados por Spades fue analizada, haciendo un alineamiento con Mauve v20150226 entre el ensamble de Spades del genoma completo y el ensamble de plásmidos efectuado por PlasmidSpades. Para el caso de la muestra SRR6514350, por ejemplo Figura 13, se puede observar cómo los *contigs* que contienen los bloques conservados en color marrón y azul, en el caso de PlasmidSpades (inferior) son más largos, posiblemente gracias a su proceso de elección de secuencias circulares y su relación compartida de profundidad, mientras Spades fragmenta estos plásmidos, ya que es menos eficiente en las regiones repetitivas del genoma.

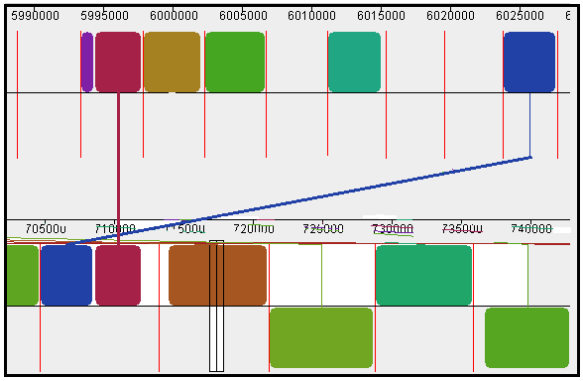


Figura 13. Mauve v 20150226, muestra 350 Spades VS PlasmidSpades

Dado que PlasmidSpades genera *contigs* distintos a los generados por Spades y que los demás clasificadores utilizan los *contigs* generados por Spades para hacer la predicción, se realizó un Blast ente los *contigs* de PlasmidSpades y los *contigs* ensamblados por Spades, con el objeto de identificar cuáles de estos correspondían y utilizar los identificadores *de* Spades como referencia en los pasos posteriores. A *priori* las secuencias de PlasmidSpades deberían estar representadas en el ensamble genómico con una identidad del alineamiento cercana al 100% pero una cobertura del *query* o subject reducida. Por esta razón en la configuración Blast del archivo de salida, se almacenó el identificador del *query* (qseqid), el identificador del subject (sseqid), porcentaje de identidad del hit (pident), los 3 primeros mejores hits de cada *contig* del *query* con la opción –Mhps de Blast, para poder evaluar aquellos alineamientos donde un *contig* de PlasmidSapdes con varios de Spades. Además se configuraron también los parámetros: tamaño del

alineamiento (Length) y tamaño del Subject (Slen), para poder determinar el cubrimiento del alineamiento.

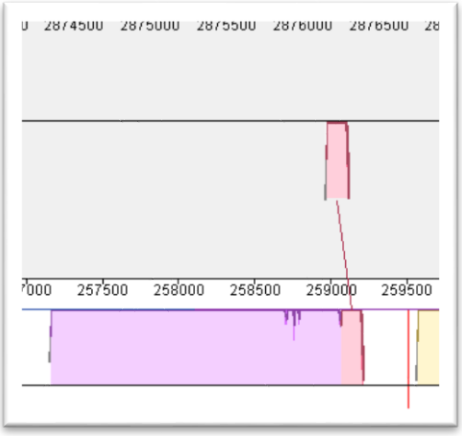


Figura 14. Match de zona repetitiva ensamblada por PlasmidSpades (inferior) contra el genoma de Spades (Superior)

La equivalencia entre los *contigs* de PlamidSpades y Spades utilizando Blast, requirió para su parametrización de una referencia, por lo que se generó un archivo con aquellos *contigs* de Spades correspondientes a los de PlasmidSpades (Anexo A), obtenido de la revisión manual de los alineamientos entre los dos set de contigs, alineamientos obtenidos utilizando la herramienta Mauve v 2015-02-26. A través de este procedimiento se excluyen aquellos hits con una alta identidad, pero una cobertura baja, como es el ejemplo representado en la **¡Error! No se encuentra el origen de la referencia.** Figura 14, donde una parte de un *contig* del ensamble de PlasmidSpades alinea dentro de un *contig* Spades, que por su tamaño es evidente que es parte de un cromosoma.

Para la parametrización se seleccionaron los siguientes rangos: porcentaje de identidad entre 90 y 100%, debido a que se buscan secuencias idénticas, pues la información genética que se espera identificar viene de las mismas lecturas de secuenciación; y porcentaje de cobertura entre 20 y 100, límite inferior deducido por una evaluación de la calidad de las métricas que mostró en bajos porcentajes de cobertura (menores a 20%) bastantes errores (Figura 15), afectando el cálculo de un modelo en estos rangos de decisión bajos. En este experimento, el script generó como salida tres archivos de texto

plano, uno para cada métrica, donde se observa el desempeño de la selección en cada una de las diferentes configuraciones dé %Identidad y %Cobertura (Anexo D).

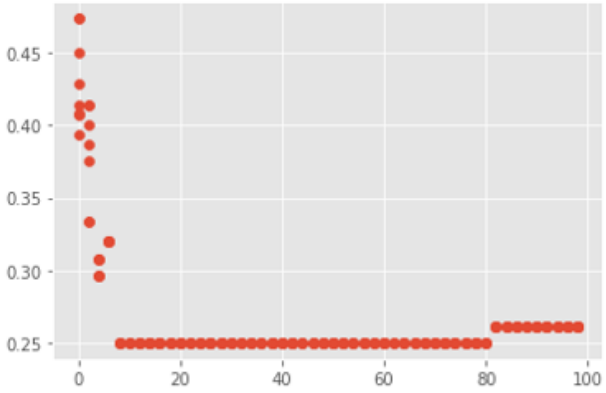


Figura 15. Porcentaje de Error en función de la cobertura, Muestra ERR2929690

Una vez se tiene el comportamiento de las configuraciones de selección de hits por cada métrica, se procedió a modelar y optimizar el comportamiento de cada métrica en función del porcentaje de Identidad y porcentaje de cobertura, con la implementación de las librerías de SKlearn de Python. Este código creado en lenguaje Python con la plataforma NootBook de Google Colaboratory (Anexo E), utiliza un modelo de dos variables de orden 2, resultando en una ecuación de 6 coeficientes para cada muestra, en cada métrica. Estos modelos se evalúan con la comparación de 33% de los datos iniciales, separados desde el inicio como datos de prueba con el uso de la función "`train_test_split`" de SKlearn. Luego con la implementación de la librería "`PolynomialFeatures`" se modeló el comportamiento de cada muestra en una ecuación de dos variables independientes (ID, Cov), permitiendo entonces con ayuda de la librería "`optimize`" optimizar y encontrar aquella configuración de porcentaje de identidad y cobertura que maximizan el desempeño de la elección de *contigs* en función de las métricas de Precisión, *Recall* y *F-Score*. En la Figura 16**¡Error! No se encuentra el origen de la referencia.**, se plasma en el eje (y) la medida porcentual óptima de elección para (a) % Identidad (azul) y (b) % cobertura (rojo), mientras que en el el eje (x) tiene en cada punto la etiqueta de la muestra correspondiente.

Las líneas azules indican los mínimos aritméticos de cobertura e identidad respectivamente.

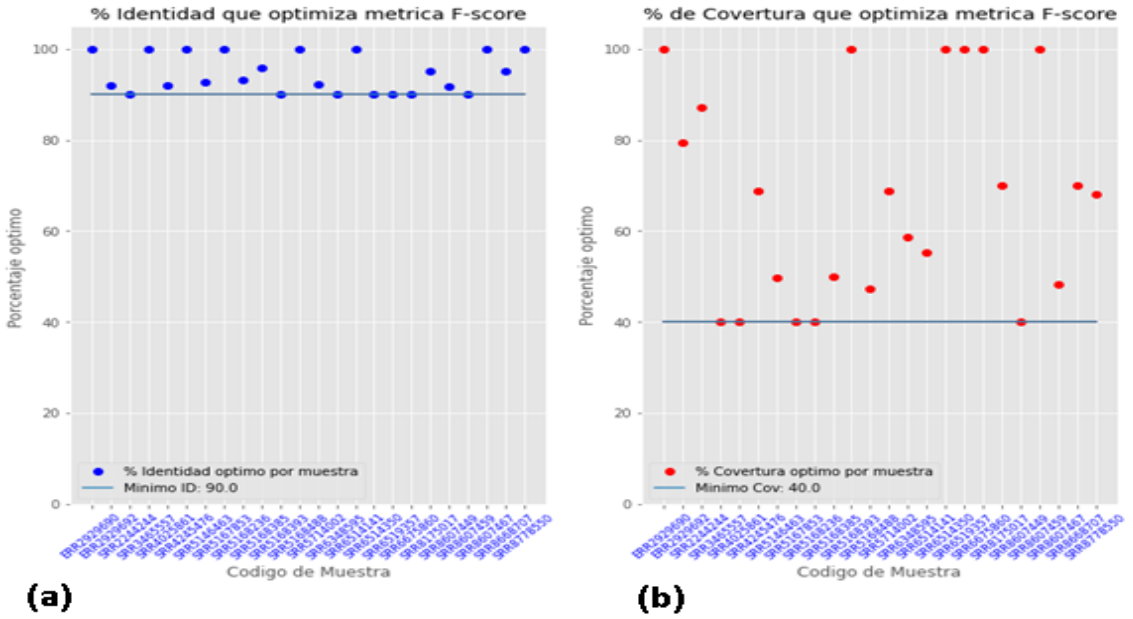


Figura 16. Gráfico de valores de decisión óptimos, Métrica *F-Score*. (a) % Identidad, (b) % Cobertura

Debido a que cada muestra tiene una procedencia, librería y profundidad distintas, se esperaba que los modelos resultado de estos ensambles sean también variables, que se evidencia por la variabilidad en las configuraciones de parámetros optimas en las distintas muestras, en especial en el *F-Score* tomado como referencia para la determinación de los parámetros, pues el cálculo de este valor es dependiente tanto de la precisión y de la exhaustividad o "*recall*". Con el análisis gráfico, basado en Mauve v 20150226 se encontró lo sensible que es Plasmidspades a etiquetar zonas repetitivas como plásmidos (Fig 13, Fig 14, pues debido a que uno de los criterios de selección es la profundidad de las secuencias, el algoritmo selecciona a los posibles plásmidos y las zonas repetitivas del genoma, porque estas tienen una profundidad mayor al promedio, lo que genera estos falsos positivos y disminuye la efectividad con la que se seleccionan las secuencias relacionadas de los ensambles. Se evidenció también que esta selección de *contigs* representados en los ensambles es afectada también por la calidad de los datos, por ejemplo, la muestra 236, tiene el peor desempeño en la elección de *contigs* representados pues los ensambles de Spades y PlasmidSpades son también los más fragmentados del

conjunto, resultando en muchos *contigs* cortos con una alta posibilidad de hacer match a lo largo de los ensambles. Teniendo en cuenta el comportamiento de los límites inferiores óptimos de decisión Figura 16, se tomaron estos como límite de decisión; con esta configuración el modelo tiene un desempeño en Precisión, *Recall* y *F-Score* como se muestra en la Tabla 8.

Tabla 8. Clasificación PlasmidSpades, todo el set de datos.

PlasmidSpades	***Precision***	***RECALL***	***F-SCORE***
Evaluado en todo el set de datos.	0.798	0.805	0.801

Este desempeño demuestra una confianza de clasificación general del 80%, esta métrica es afectada principalmente por dos razones:

- La presencia de información genética con una profundidad de nucleótido relativamente alta, debido en su mayoría a la presencia de zonas repetitivas del genoma y los procesos como fragmentación y amplificación, necesarios en la metodología de secuenciación NGS.
- El proceso de unificación de identificadores de Spades con los alineamientos Blastn, afecta negativamente el desempeño de clasificación, pues es de esperar falsos negativos en el filtrado de estos hits.

Clasificación por profundidad de Nucleótido relativa

Una de las características de los plásmidos es su tasa de replicación, la cual es mayor a la del genoma bacteriano, esta característica deja una huella en las lecturas producto de una secuenciación NGS, ya que los plásmidos que tengan replicas al momento de secuenciar, tendrán un numero de lecturas proporcional al número de estas copias. Esta característica tiene un efecto también en el Ensamble *de novo,* pues los *contigs* ensamblados con estas lecturas redundantes terminan adquiriendo una profundidad promedio de *k-mer* (en el caso de Spades) mayor a los *contigs* provenientes del cromosoma. Pero debido al proceso fisicoquímico de secuenciación, donde se fractura y

amplifica la información genética, resultan en el ensamble *contigs* que acumulan las zonas repetitivas del genoma, dichos *contigs* también tienen por consiguiente una profundidad de *k-mer* mayor al resto del cromosoma (código relacionado en Anexo F).

El script en Python implementado calculó la profundidad de nucleótido de cada *contig* del ensamblaje en función de la profundidad de k-mer del encabezado de cada *contig* generada por Spades, a partir de los archivos de ensamble de Spades (Muestra*.filter1000.fasta), generando tablas para aislamiento con el resultado de la clasificación de cada *contig*, en donde columnas corresponden a *Contig* ID, Cobertura de Nucleótido y Tipo de Secuencia (Cromosoma 0, Plásmido 1) (Figura 17**¡Error! No se encuentra el origen de la referencia.**a). Se genero una tabla por cada uno de los 100 límites de selección evaluados en el rango de 1,5 a 3 calculados utilizando la ecuación 1, que van desde el valor de VA hasta el de VEA. A partir de estas tablas se calcularon para cada limité los valores de Precisión, Recall y F-score, lo cuales fueron integrados en una tabla (Figura 17b) donde los valores en las columnas muestran el desempeño en función del límite de decisión “porcentaje”, primera columna, y en las siguientes columnas se tiene el desempeño en cada muestra del Set de datos.

(a)

ERR2929690_ID	Nucle_cov	Tipo_sec
51.0	39.0	0.0
52.0	63.0	1.0
53.0	35.0	0.0
54.0	15.0	0.0
55.0	48.0	1.0

(b)

Porcentaje	ERR2929690	ERR2929692	SRR2244244
1.500	0.428571	0.320000	0.225806
1.515	0.428571	0.320000	0.225806
1.530	0.428571	0.320000	0.225806
1.545	0.428571	0.320000	0.225806
1.560	0.428571	0.320000	0.225806
1.575	0.428571	0.320000	0.225806

Figura 17. Tabla profundidad de Nucleótido.

Una vez se tiene el comportamiento de eficiencia de clasificación del modelo para todas las muestras, se procede a modelar el comportamiento en función del límite de decisión, esto implementando las librerías “PolynomialFeatures” de Skalearn y “sm” de Statsmodels ver AnexoF; Con una ecuación que describe el comportamiento de la clasificación en función del límite de decisión ver Figura 18, es posible entonces calcular el valor del límite

que optimiza cada muestra, utilizando la librería “optimize” de “Scpy” para calcular dentro de los limites (1.5-3.0) el valor que maximiza la eficacia de clasificación.

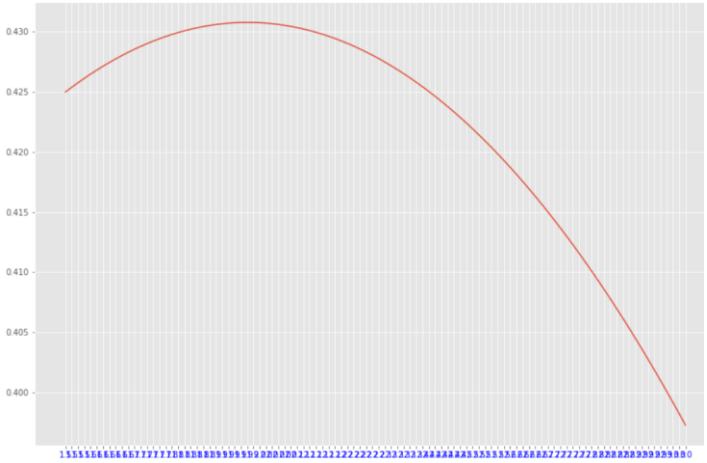


Figura 18 Función representativa de la eficacia de clasificación respecto al límite de decisión (1.5-3.0), para la muestra ERR2929690.

Fue posible evidenciar sobre las funciones independientes de cada muestra y el desempeño general, fue lineal entre el valor atípico y extremadamente atípico, este comportamiento es evidenciable en Tabla 9, donde se detalla el promedio del valor minimo(Atipico) y máximo (Ext, Atipico), en las métricas de evaluación Precision, Recall y F-Score, el valor estadístico que mejor cumple la función de límite de decisión, es VA(Valor Atípico) es decir clasificar a un *contig* del ensamble *de novo* como plásmido, si su profundidad de nucleótido es mayor al *Outlayer* (VA) Ecuación 2, definido anteriormente.

Tabla 9. Desempeño modelo Atípico, Extremadamente Atípico

VALOR LIMITE	***PRECISION***	***RECALL***	***F-SCORE***
EXTREMADAMENTE ATÍPICO	0,5149	0,1654	0,2269
ATÍPICO	0,5571	0,2631	0,3324

Clasificación por Similitud de secuencias, con Blastn

Base de Datos

Se implementó un script en Python con base en. Diagrama de flujo de los procesos secuenciales necesarios para generar las bases de datos Blast de cromosomas de la especie bacteriana y plásmidos Figura 19, necesarias para hacer la clasificación de contigs por similitud de secuencia.

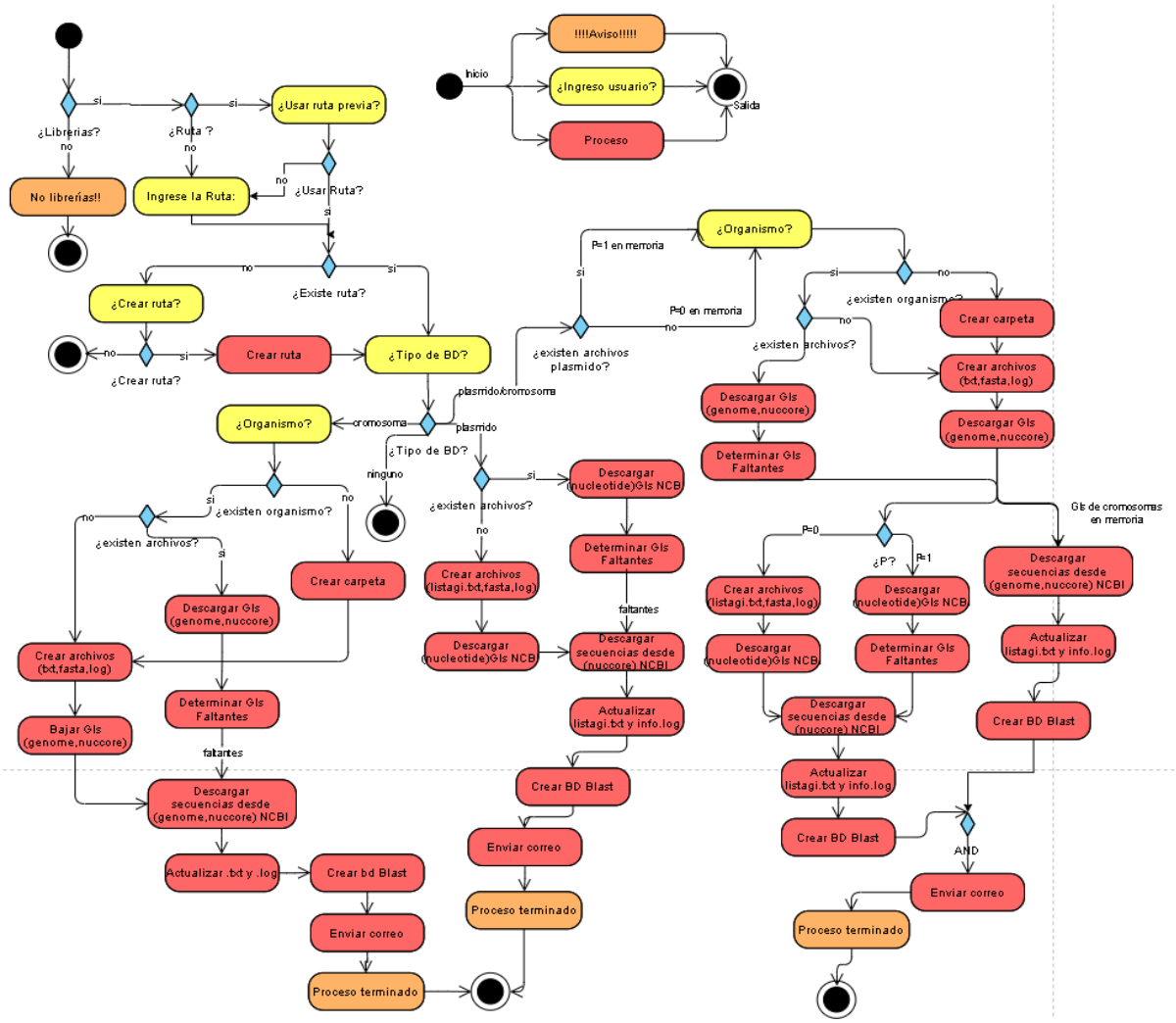


Figura 19. Flujograma Create_Update_DB.py

En este en este diagrama de flujo inicialmente se procede a establecer una ruta donde se almacenarán las carpetas contenedoras de las bases de datos de cromosomas y de plásmidos en el sistema de almacenamiento del servidor Linux, el script trabaja con una estructura de datos como se esquematiza claramente en Figura 20, para cada base de datos en la carpeta principal se establecen los siguientes archivos:

- **(plasmidos/cromosomas).fasta**: Archivo multi-fasta con las secuencias de nucleótidos e identificadas por Acces Numbers del NCBI.

- **ListaGI.txt:** Archivo de texto que contiene la lista de Unique GenBank Numbers de las secuencias descargadas, estas separadas por un espacio en blanco tipo tab '\t'.
- **Info.log:** Archivo de seguimiento de la creación y actualización de las bases de datos, allí reposa el conteo de secuencias almacenadas y el tiempo de inicio y fin de cada tarea efectuada.
- **(plasmidos/cromosomas)DB**: carpeta que contiene los archivos generados por el software "makeblastdb", script que genera la base de dataos tipo Blast para el posterior alineamiento local múltiple de las secuencias sin identificar.

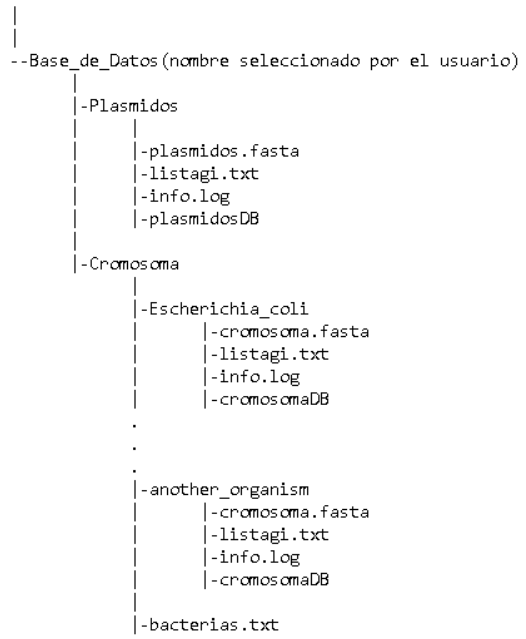


Figura 20. Estructura Base de Datos

Luego de que el script salga satisfactoriamente del proceso de definición de la ruta principal el programa según el modo de operación (Interactivo, línea de comando) dará inicio a una de las tres tareas definidas en el diagrama de flujos Figura 19. Estas tareas son:

1. Crear o Actualizar base de datos de plásmidos.
2. Crear o Actualizar base de datos de Cromosoma.
3. Crear o Actualizar base de datos de plásmidos y de Cromosoma.

En las tres funciones, el usuario debe mínimo debe especificar el nombre del microorganismo que desea incluir en su base de datos. Las bacterias cuyas base de datos

se han creado y están disponibles para pasos posteriores son almacenados el archivo de la carpeta cromosomas llamado “bacterias.txt” Figura 20**¡Error! No se encuentra el origen de la referencia.**, archivo que es consultado por el script para saber si el proceso actual es de creación o actualización de los archivos de una bacteria. El usuario también decide si desea ser notificado vía email, debido a que el proceso de descarga de las secuencias suele tomar grandes cantidades de tiempo, lo que hace recomendable además, que en la opción de línea de comando el script de ejecute en segundo plano y en la opción interactiva una vez comience la descarga, pasar a segundo plano con las funciones (crt+z) y fg %<#>, donde # es el código de la operación asignada en segundo plano.

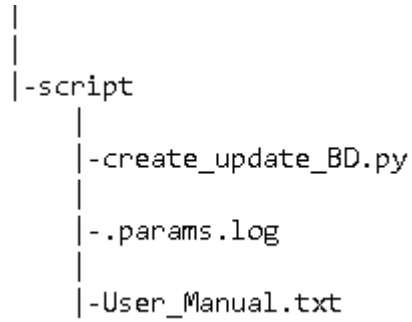


Figura 21. Estructura directorio raíz.

La implementación del diagrama de flujo incluye dos archivos en el directorio raíz del script: .params.log y User_manual.txt Figura 21**¡Error! No se encuentra el origen de la referencia.**. “.params.log” es un listado de las rutas principales de bases de datos que han sido creadas en el sistema Linux local, las cuales pueden ser vistas y seleccionadas desde la opción de uso interactivo del script o con la opción de línea de comando, abriendo el archivo con un editor de texto para verificar la ruta. El objetivo principal de este archivo es evitar redundancias en la creación de bases de datos, ya que pueden afectar la capacidad de almacenamiento del Servidor Linux. Por otro lado el archivo “User_Manual.txt“, es accedido por línea de comando con la opción (-h) para instrucciones de uso o (-m) para ver todo el archivo con descripción, funcionamiento, estructura de datos, librerías requeridas y versión.

Las bacterias Recopiladas junto con los *contigs* plasmídicos son los detallados en la Tabla 10, donde se evidencia la cantidad de secuencias como el tiempo de ejecución del script, valor proporcional a la cantidad y tamaño de las secuencias.

Tabla 10 Resultado de Ejecución Script BD

Tipo de Secuencia	Descripción	No de secuencias	Tiempo de Ejecución (min)
Plásmidos	Enterobacter	116544	75
Cromosoma	Klebsiella pneumoniae	100	18
	Providencia rettgeri	35	6
	Pseudomonas aeruginosa	24	4
	Acinetobacter baumannii	1491	59

Clasificación por similitud de secuencias

La estructura funcional del proceso de clasificación de contigs por homología frente a una base de datos de plámidos y a una de cromosomas de *K. pneumoniae* **¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.**, tiene como entrada los ensambles de Spades en formato multifasta con los que se hacen alineamientos Blastn, contra una base de datos específica. En un caso se seleccionan los *contigs* que tienen secuencias homologas en la base de datos de plásmidos y en el otro caso se seleccionan como plásmidos aquellos *contigs que* no tienen buenos alineamientos con la base de datos de cromosomas de *K. pneumoniae*.

Tabla 11. %ID, %COV, *F-Score* Similitud DB Plásmidos

ID	COV	ERR2929690	ERR2929692	SRR2244244	SRR3465532
80.0	20.0	0.4000	0.3678	0.5818	0.1818
80.0	21.0	0.4048	0.3678	0.5818	0.1818
80.0	22.0	0.4048	0.3678	0.5818	0.1905
80.0	23.0	0.4048	0.3678	0.5818	0.1905

Los rangos en los que se modela el comportamiento de la métrica *F-Score* según la configuración de (%ID, %QCOVS) son diferentes para los dos métodos, pues en uno se buscan secuencias similares como lo es el caso de plásmidos o buscar secuencias diferentes a las de la base de datos, como es el caso de cromosomas.

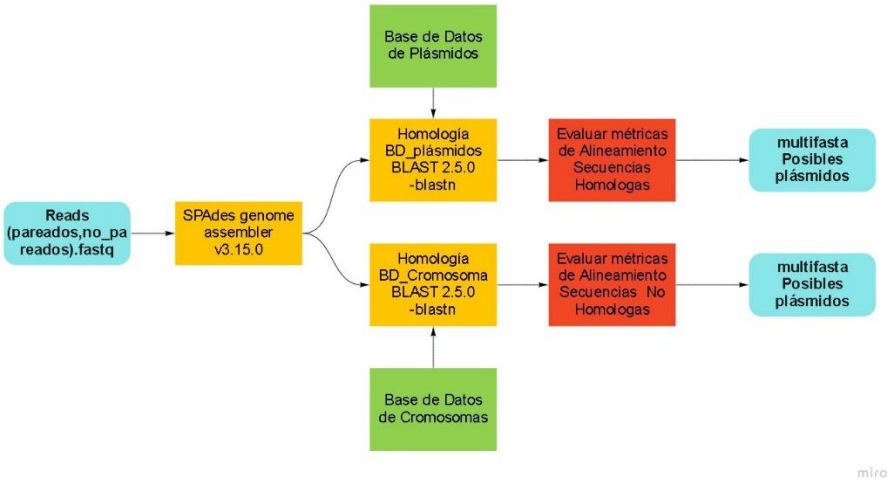


Figura 22. Clasificación por similitud de secuencia

Para cada método (Similitud y No-Similitud) se genera un archivo con métricas de alineamiento en una tabla con formato separado por tabs “\t”, producto de la ejecución de Blastn, dichas tablas son accedidas y unificadas con un script en Python, que en base a la referencia de plásmidos, explora y califica la clasificación del método en un rango de configuración de límites de discriminación %ID y %Cov ver Tabla 11**¡Error! No se encuentra el origen de la referencia.**, donde las dos primeras columnas representan las

variables independientes y las demás representan la variable dependiente *F-Score* para cada muestra, dichas tablas accesibles en Anexo E.

Una vez se tiene el comportamiento de eficiencia de clasificación del modelo para todas las muestras, con el algoritmo descrito previamente y ejecutado en “Google Colab” fue posible evidenciar sobre las funciones independientes de cada muestra y el desempeño general, que los límites para la selección se contigs plasmídicos fueron de porcentaje de identidad mayor que 99% y cobertura del “*query*” mayor a 60% Figura 23**¡Error! No se encuentra el origen de la referencia.**.

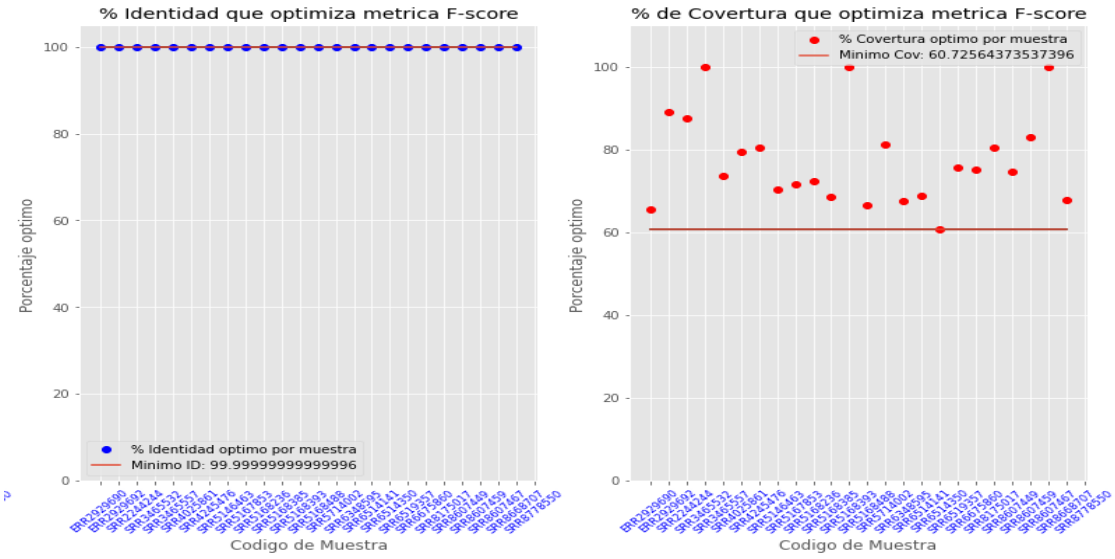


Figura 23. Evaluación de porcentajes de decisión óptimos, para alineamiento contra BD de plásmidos.

Para el caso de la selección de *contigs* posibles plásmidos a partir de una No-Similitud contra la base de datos de cromosomas como se ve en Figura 24, se encontraron los limites superiores de decisión 99% de identidad y 93% de cobertura, es decir este método de selección determina a un posible plásmido si no tiene un alineamiento muy bueno contra los cromosomas de la Base de Datos.

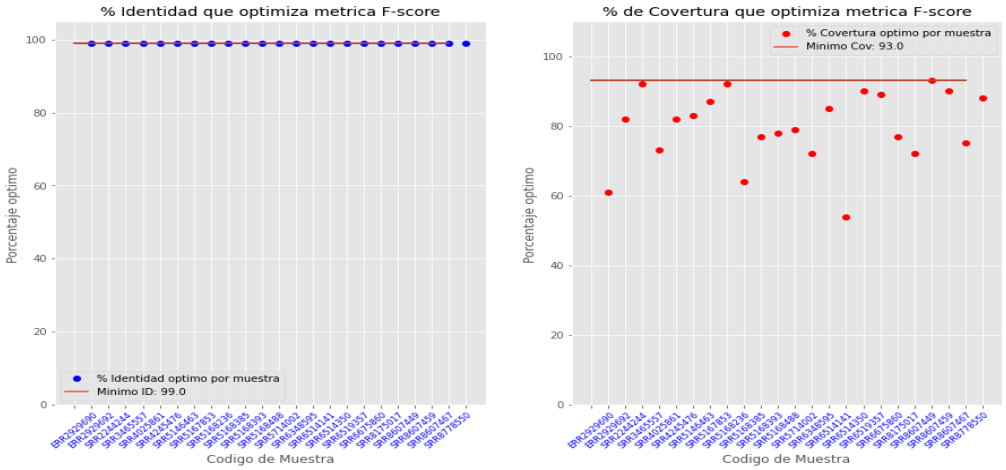


Figura 24. Evaluación de porcentajes de decisión óptimos, para alineamiento contra BD de Cromosomas.

El desempeño de estos dos métodos es evaluado con todo el set de datos, usando los límites de decisión recién mencionados, este desempeño se detalla en la Tabla 12, donde el alineamiento contra base de datos de plásmidos resulta ser el método más específico de los hasta ahora evaluados con casi 100% de *recall,* a diferencia de la selección por no-similares que sobre el conjunto de las secuencias plásmidicas puede identificar solo una cuarta parte de los plásmidos con 25% de *F-Score*, 17% de *recall*, además los *contigs* clasificados por este método, más de la mitad son falsos positivos con 48% de *precisión, es decir identifica como plásmidos secuencias de cromosoma cuyos límites de decisión son compartidos con los buscados para optimizar la discriminación de plásmidos.*

Tabla 12. Métodos Similitud de Plásmidos y No-similares Cromosomas

Método	***Precision***	***Recall***	***F-Score***
Similitud de BD Plásmidos	0.783	0.998	0.878
No-Similitud BD Cromosoma	0.484	0.174	0.256

Evaluación Herramienta PlasFlow

Esta evaluación se realizó del mismo modo que las herramientas anteriormente evaluadas, comparando los *contigs* que PlasFlow v.1.1 clasifica como plásmidos, contra los *contigs* de referencia obtenidos con Mauve y las secuencias reales de plásmidos y cromosomas. Esta evaluación se realizó de la misma forma que los métodos anteriores usando el archivo de texto plano generado por PlasFlow v.1.1 para cada uno de los ensambajes de Spades, sobre los que se calcularon precisión, Recall y F-score, y el comportamiento general de desempeño Tabla 13**¡Error! No se encuentra el origen de la referencia.**. El resultado evidencia el potencial de la aplicación de métodos de inteligencia artificial, para la identificación de patrones propios de secuencias de plásmidos, ya que cuenta con un desempeño de clasificación para los dos tipos de secuencias (plásmidos, cromosoma), con 75% de F-score, el mejor después del método de similitud de secuencias contra bases de datos específicas de plásmidos. La fortaleza de esta metodología de clasificación es que permite la identificación de secuencias plasmídicas que no hayan sido previamente reportadas como plásmidos.

Tabla 13. Desempeño PlasFlow v.1.1

Método	***Precision***	***Recall***	***F-Score***
PlasFlow v.1.1	0,7018	0,8134	0,7503

Las métricas de clasificación son únicas para PlasFlow, ya que esta herramienta no cuenta con parámetros para variar su desempeño.

Multi-Clasificador

A continuación, se presenta la unificación de los diferentes experimentos realizados para optimizar los parámetros de configuración de cada una de las herramientas involucradas en el Multi-Clasificador general de plásmidos Figura 25. Además, se muestran los resultados de las evaluaciones de distintos conjuntos de clasificadores débiles, junto a la evaluación de variados algoritmos de unificación de resultados.

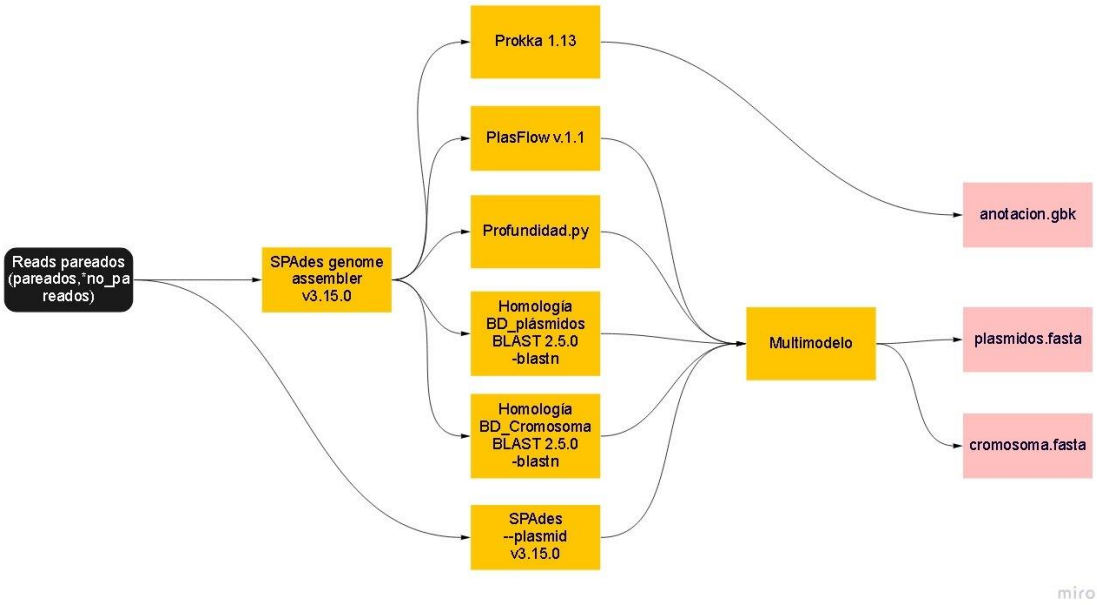


Figura 25 Pipeline para Clasificación y Anotación de secuencias de Plásmidos y Cromosoma

Para la unificación de resultados de los diferentes clasificadores débiles optimizados, se evaluaron los siguientes algoritmos multi-clasificador:

- *Bagging*, *Hard vote*: Estas aproximaciones son las más básicas y equivalentes pues las dos no requieren de un entrenamiento, pues clasifican promediando las salidas de los clasificadores débiles en el caso de *Bagging* o dándole un mismo peso en la decisión final a cada clasificador débil como es el caso de *Hard vote*.

- *Boosting*: Esta aproximación requiere de un entrenamiento estructurado en serie que busca encontrar un arreglo de pesos que reduzcan el error general del multi-clasificador en función del error de cada clasificador débil, ya que es un entrenamiento en serie, su resultado será dependiente del orden de los clasificadores débiles, lo que limita su implementación por el espacio de búsqueda que aumenta dramáticamente con la cantidad de clasificadores débiles implementados.
- *Staking* (SVM): Este método de inteligencia artificial ampliamente usado, extrapola el vector de entrada a un espacio característica lo que permite tener fronteras de decisión entre las clases más complejas a diferencia de un método lineal de clasificación.
- *Soft Voting*: Aproximación estadística que involucra el desempeño previo de cada clasificador débil como un peso, dándole entonces prioridad a la clasificación de aquellas herramientas que en un experimento se comportó mejor en relación a los demás.

Tabla 14 Combinaciones posibles de 5 Clasificadores débiles.

BD_plasmidos,BD_No_cromosoma
Plasflow,BD_No_cromosoma
Plasflow,BD_plasmidos
Plasflow,Profundidad
Plasmidspades,BD_No_cromosoma
Plasmidspades,BD_plasmidos
Plasmidspades,Plasflow
Plasmidspades,Profundidad
Profundidad,BD_No_cromosoma
Profundidad,BD_plasmidos
Plasflow,BD_plasmidos,BD_No_cromosoma
Plasflow,Profundidad,BD_No_cromosoma
Plasflow,Profundidad,BD_plasmidos
Plasmidspades,BD_plasmidos,BD_No_cromosoma
Plasmidspades,Plasflow,BD_No_cromosoma
Plasmidspades,Plasflow,BD_plasmidos
Plasmidspades,Plasflow,Profundidad
Plasmidspades,Profundidad,BD_No_cromosoma
Plasmidspades,Profundidad,BD_plasmidos
Profundidad,BD_plasmidos,BD_No_cromosoma
Plasflow,Profundidad,BD_plasmidos,BD_No_cromosoma
Plasmidspades,Plasflow,BD_plasmidos,BD_No_cromosoma
Plasmidspades,Plasflow,Profundidad,BD_No_cromosoma
Plasmidspades,Plasflow,Profundidad,BD_plasmidos
Plasmidspades,Profundidad,BD_plasmidos,BD_No_cromosoma
Plasmidspades,Plasflow,Profundidad,BD_plasmidos,BD_No_cromosoma

Este experimento evalúa el *F-Score* de 130 distintas configuraciones de modelo y el conjunto de clasificadores (Tabla 14), métrica que representa de manera general el

desempeño de los multi-clasificadores respecto a *presicion* y *recall*. En la Figura 26 se muestran estos resultados, en donde en el eje (x) o frontal de la vista se tienen las 26 posibles configuraciones de clasificadores débiles ver Tabla 14, en el eje (y) los 5 distintos algoritmos de unificación de resultados, siguiendo el orden a continuación:

1. *Bagging*
2. *Boosting*
3. *Staking*
4. *Hard vote*
5. *Soft Vote*

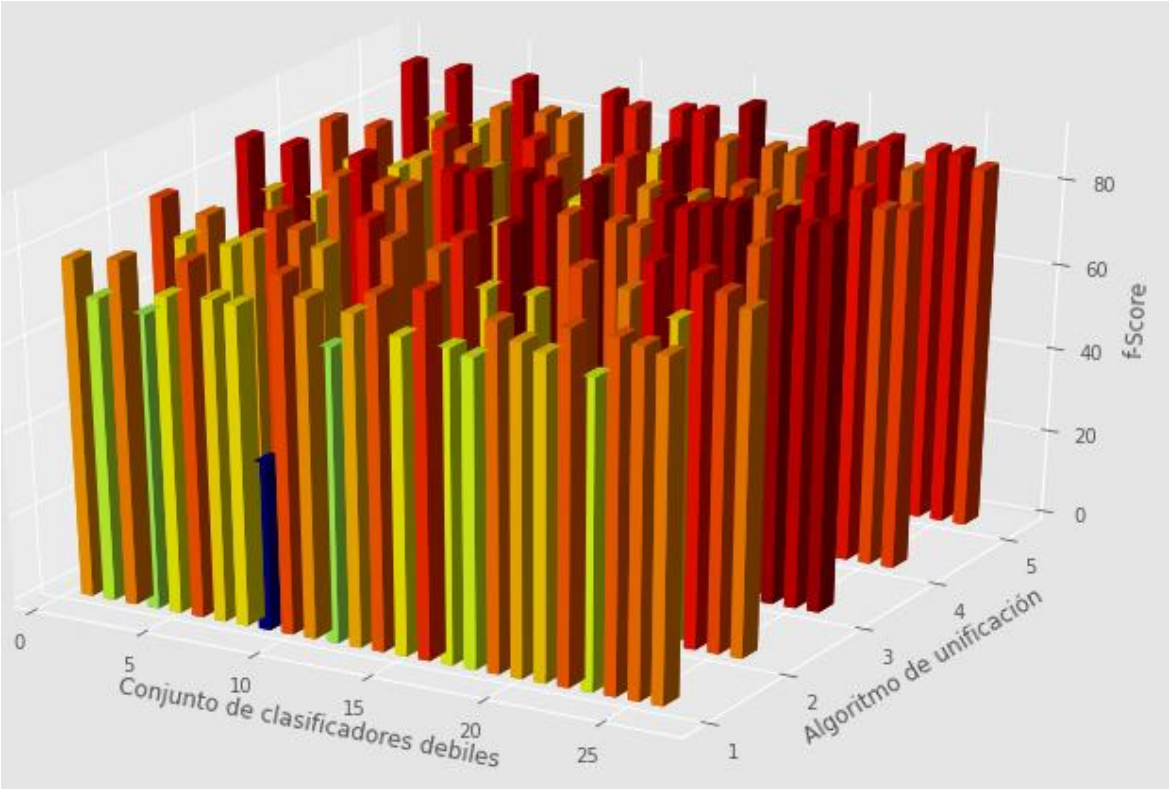


Figura 26. *F-Score* para Multi-clasificadores VS Conjuntos de clasificadores débiles.

Para cada uno de los cinco algoritmos de unificación se identificó el conjunto de clasificadores débiles con mejor desempeño Tabla 15, donde es importante resaltar, como

para la mayoría de algoritmos el conjunto con mejor desempeño es (Plasmidspades, PlasFlow, BD similitud plásmidos), a diferencia del método "*Staking*" con el conjunto (Plasmidspades, PlasFlow v.1.1, Profundidad, BD similitud plásmidos, BD No-similares Cromosoma) que por cierto también es el que mejor se desempeña con 91.23% en la métrica *F-Score*.

Tabla 15. Comparaciones Algoritmos de unificación con el mejor conjunto de clasificadores débiles.

Algoritmo de unificación	**Conjunto con mejor desempeño**	***F-Score* %**
Bagging	Plasmidspades/PlasFlow /BD_plasmidos	85.64
Boosting	Plasmidspades/Plasflow/BD_plasmidos	85.65
Staking	Plasmidspades/Plasflow/Profundidad/BD_plasmidos/BD_No_cromosoma	91.23
Hard vote	Plasmidspades/Plasflow/BD_plasmidos	85.63
Soft Vote	Plasmidspades/Plasflow/BD_plasmidos	74.33

El desempeño del multi-clasificador seleccionado para el Pipe Line *Staking,* es descrito en detalle en la Tabla 16, donde muestra en cada fila el desempeño según las métricas *precisión, recall y F1-Score*, para cada clase (cromosoma y plásmidos) y la calificación general del multi-clasificador en la tercera fila. Es importante resaltar que para el multi-clasificador Staking es más sencillo clasificar secuencias provenientes del cromosoma, que evaluando secuencias Plasmidicas, debido a la diferencia en sus desempeños por clase.

Tabla 16. Evaluación PipeLine contra todo el set de datos.

	Precision	Recall	F1-Score
Clasificación de Contigs Cromosomales	0,966	0,916	0,94
Clasificación de Contigs de Plásmidos	0,84	0,932	0,884
Calificación general de Clasificación	0,903	0,924	0,912

Pipeline

Análisis de requerimientos:

A. Administración de ficheros, para la organización de los procesos y sus resultados en el sistema de archivos definido, Librería de Python “sys”, para la interacción con ficheros desde lenguaje de programación con sistema operativo Linux.
B. Acceso deliberado a variados ambientes Linux, pues en la implementación de variadas herramientas es usual requerir de paquetes y versiones de librerías específicas.
C. Diseñar un sistema de administración de procesos que cuente con estabilidad de ejecución y con la capacidad de ejecutar procesos en segundo plano.
D. Capacidad de sobrellevar errores y evitar caídas generalizadas en un conjunto por errores singulares de una muestra.
E. Función inscrita de administración de archivos de recopilación de ejecuciones, detalles y cronometrado de tareas.
F. Función para el reconocimiento de procesos incompletos para actualizar ejecuciones y evitar rehacer tareas en una muestra con previa ejecución.

G. Función para ejecutar script ‘hijo’ ”Create_Update_BD.py” de actualización de bases de datos, en caso que el usuario desee hacer una actualización o agregar un nuevo organismo en la base de datos local.
H. Capacidad de informar al usuario el estado de ejecución vía correo electrónico.
I. Opción de ejecutar el script desde línea de comandos y por consola interactiva.
J. Guardado de directorios previos usados, para acelerar los procesos de ejecución del script con un menú informativo y de fácil interacción.
K. Instalación Software Spades, PlasmidSpades y requerimientos específicos:
    a. g++ (version 5.3.1 or higher)
    b. cmake (version 2.8.12 or higher)
    c. zlib
    d. libbz2
L. Instalación Software Blast de NCBI, requerimiento especifico de compilador c++.
M. Instalación PlasFlow, junto a requerimientos:
    a. Python 3.5
    b. Python packages:
        i. Scikit-learn 0.18.1
        ii. Numpy
        iii. Pandas
        iv. TensorFlow 0.10.0
        v. rpy2 >= 2.8
        vi. scipy
        vii. biopython
        viii. dateutil >= 2.5
    c. R 3.25 packages: Biostrings
N. Para el cargado de modelos pre-entrenados en la unificación de resultados es necesario la librería joblib de Python.
O. Creación y lectura de archivos biológicos en formato Fasta, librería Bio de Python.
P. Instalación del software Prokka, para la anotación de los archivos fasta clasificados por el Pipelne.

Diseño Algoritmo

El proceso generalizado que ejecuta el pipeline se evidencia en Figura 27, donde se aprecia una nomenclatura de procesos, basada en colores que describen el tipo de tarea a ejecutar así:

a. Naranja: Proceso no interactivo, informativo que permite al usuario conocer de manera clara, el origen del problema.
b. Rojo: Proceso Informático de ejecuciones de software accesorios del Pipeline Bioinformático, son también tareas que pueden o no, ser ejecutadas en segundo plano.
c. Amarillo: Proceso interactivo de establecimiento de parámetros de ejecución, pueden ser establecidos por línea de comando o por menú interactivo.

Este Pipe line a partir de las secuencias limpias de secuenciación, ensambla con Spades y con PlasmidSpades, con los identificadores de los *contigs* de Spades Clasifica las secuencias por Similitud de plásmidos, No-similitud Cromosoma, Profundidad relativa, PlasFlow y PlasmidSpades. Después unifica los resultados de estos cinco métodos de clasificación con un algoritmo de inteligencia artificial SVM, para de esta manera generar tres archivos consolidados (Plámsidos, Cromosoma

y Genoma), con sus respectivas anotaciones Prokka y anotación especifica de resistencia con el software RGI.

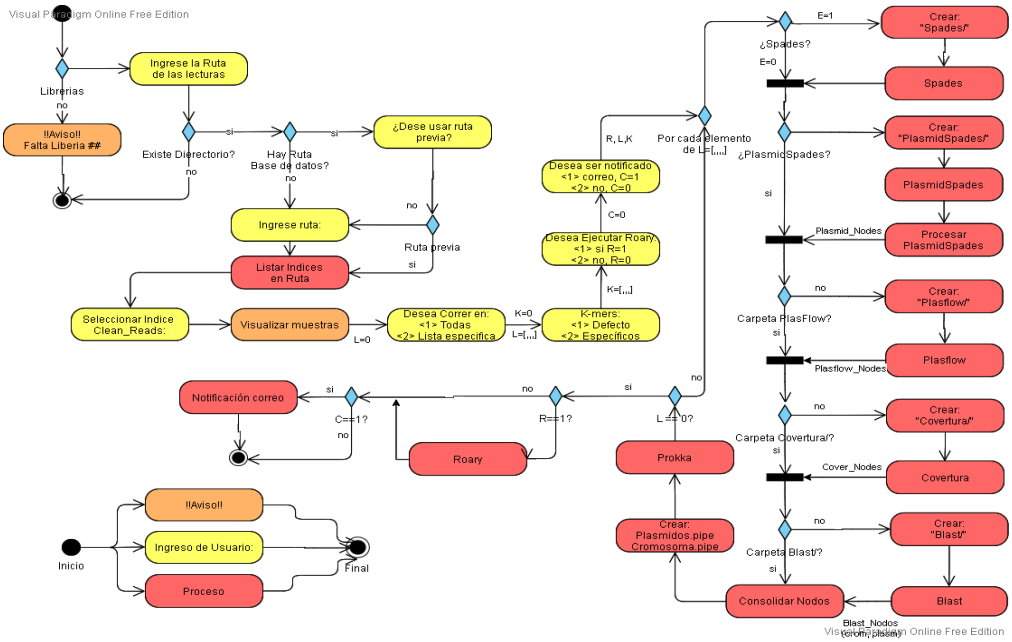


Figura 27. Diseño Flujograma PipeLine de identificación y anotación de plásmidos.

Implementación del Pipeline

La implementación del Pipeline se ejecutó en el servidor a disposición del grupo de investigación con las características especificadas en la sección Multi-clasificador, con sistema operativo Linux so distribución openSUSE Leap V15.1, implementando lenguaje bash y Python V3.7.

El script de identificación y anotación de secuencias cromosomales y plasmídicas, a partir de secuenciación de genoma completo requiere como datos de entrada una estructuración de archivos tal como se evidencia en Figura 28, donde mínimo se necesita tener una carpeta por muestra que tenga como hijo la carpeta "*Reads*_Original/", allí deben alojarse las lecturas de secuenciación NGS, con un formato que permita identificar el archivo de lecturas Forward y Reverse.

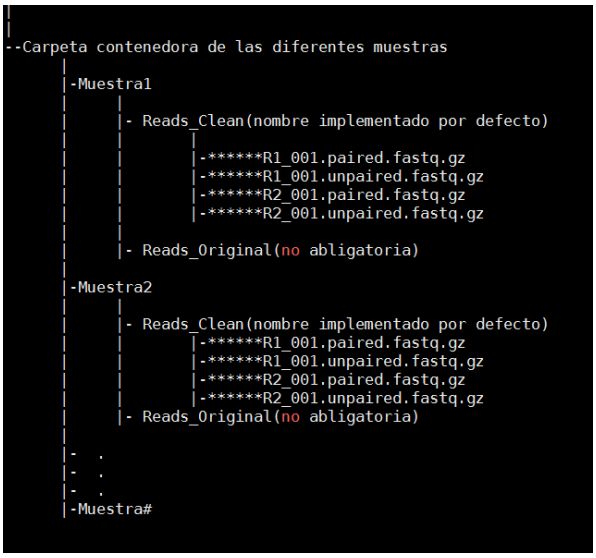


Figura 28. Estructura de archivos mínima para ejecución pipeline.

El Pipeline puede ejecutarse de dos maneras, una interactuando con el usuario hasta definir las características de ejecución ejecutar: >>Pipe_sgig.py ó segunda, por medio de una línea de comando donde se definan todos los parámetros de ejecución ejecutar: >>Pipe_sgig.py [options] {arguments}. Al momento de evaluar el presente pipe se contaba con los ficheros asociados Blast de los organismos actualizados en la base de datos local:

- Acinetobacter_baumannii
- Klebsiella_pneumoniae
- Providencia_rettgeri
- Pseudomonas_aeruginosa

El script “Pipe_sgig.py” se encuentra alojado en un repositorio público de GitHub ver **Anexo k,** junto al script de ejecución en segundo plano “An_Back.py”, más los detalles de configuración de ejecución y ayudas en el manual de usuario.

La salida del Pipeline está estructurada sobre la carpeta llamada “contenedora de las diferentes muestras”, allí se crea junto a las muestras una carpeta “QC”, ver Ilustración 29-(a), donde se alojaran los análisis de profundidad de nucleótido ponderada sobre las

lecturas limpias, este análisis es útil para evaluar la calidad del resultado de una secuenciación, pues se ponderan los análisis de todas las muestras allí contenidas en el archivo “deep_report.txt”, también se crea un archivo de información de ejecución llamado Anotation_Bacteria.log, allí se detallan los procesos y tiempos requeridos para cada subproceso para todas las muestras analizadas. Además en cada carpeta de muestra se crea una estructura de directorios como se ve en Ilustración 29-(a), en la carpeta “Annotation” se alojan las ejecuciones de anotación con Prokka y la anotación de resistencia de la herramienta RGI, en la carpeta “Assembly” se encuentran tanto el ensamblaje de Spades, como todos los archivos generados por los clasificadores débiles y los archivos intermedios necesarios para la generación de los consolidados fasta (consolidated_chrom.fasta, consolidated_plasmids.fasta), presentes en la carpeta de cada muestra, también se observan las carpetas Reads_Original y Reads_Clean, donde se alberga las lecturas antes y después de limpieza respectivamente.

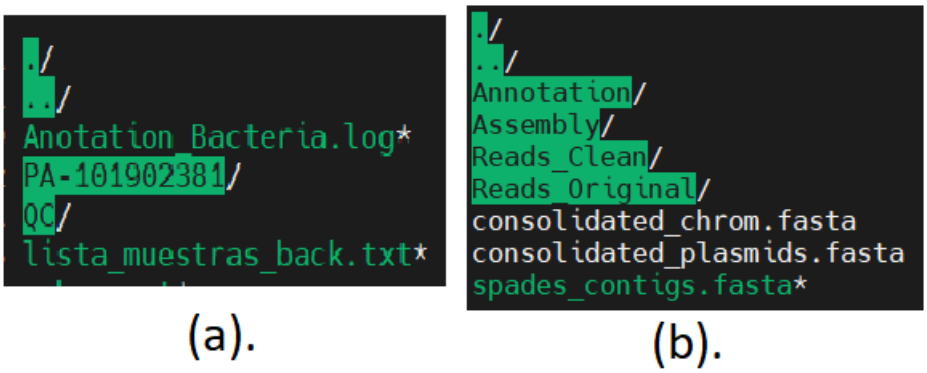


Ilustración 29 Estructura Archivos de ejecución Pipeline de Identificación y anotación de *contigs* de plásmidos y cromosoma.

El resultado de una ejecución sobre una muestra de *Pseudomonas aeruginosa*, se detalla en Tabla 1Tabla 17, donde se aprecian los procesos necesarios para correr una muestra desde los *Reads* limpios hasta obtener los archivos de anotación. Es importante resaltar que el ensamble *de novo* y la búsqueda de secuencias similares contra la base de datos de plásmidos, son los procesos más costosos desde el punto de vista de tiempo de ejecución, el primero es variable y depende de la calidad y cantidad de las lecturas; y el segunto por el tamaño de la base de Datos de plásmidos.

Tabla 17 Descripción detallada tiempos de ejecución.

Proceso	Tiempo de Ejecución (min)
Ensamble de novo Spades	00:08:10.000
Filtrado Spades	00:00:02.000
Calidad de Ensamble Quast	00:01:10.002
PlasmidSpades	00:06:53.000
Filtrado PlasmidSpades	00:00:02.000
Alineamiento PlasmidSpades/Spades	00:02:02.005
Clasificación por Plasflow	00:00:59.026
Clasificación por profundidad	00:00:14.000
No-Similitud Cromosoma	00:05:30.000
Similitud Plásmidos	00:58:30.000
Generacion tabla de consolidados	00:02:15.010
Anotacion Prokka	00:20:45.011
Anotacion RGI	00:10:03.000
Total Tiempo de Ejecución	1:28:58

Evaluación pipeline frente a secuencias de *Pseudomonas aeruginosa*.

Una vez ejecutado todo el pipeline con el nuevo set de datos, sus resultados son contrastados con los esperados, implementando un script en Python y la librería "*classification_report*" del paquete *SKlearn.* El modelo se evaluó como se muestra en la Tabla 18, donde se aprecian en las filas los desempeños con relación a los aciertos en cromosoma, plásmido y calificación del modelo balanceado (AnexoI).

Tabla 18. Calificación PipeLine con especie P. aeruginosa.

	Precision	Recall	F1-Score
Clasificación de Contigs Cromosomales	0,996	0,953	0,974
Clasificación de Contigs de Plásmidos	0,275	0,826	0,413
Calificación general de Clasificación	0,635	0,889	0,693

En esta evaluación se aprecia un decaimiento en la calificación *F-Score* del desempeño general del modelo, alrededor del 20% con referencia a la evaluación *Klebsiella pneumoniae*. Es importante resaltar que esta ejecución fue revisada usando aquellos *contigs* que el modelo clasificó mal, con alineamientos Blastn contra las bases de datos del NCBI, donde se encontró que estos *contigs* tenían secuencias de ADN que han sido reportadas tanto en cromosomas como en plásmidos, lo que explica de manera objetiva que los comportamientos biológicos de transferencia de genes entre los plásmidos al cromosoma, hacen del problema de clasificar estos dos tipos de secuencias una tarea más compleja, partiendo de secuenciación NGS.

El uso de este flujo de trabajo o Pipeline, es una buena opción si únicamente se cuenta con secuenciaciones de nueva generación o “High throughput sequencing”, ya que aprovecha varias metodologías de clasificación, para buscar un resultado más confiable, Los datos demuestran que el multi-modelo Staking mejoró el desempeño relativo sobre el mejor de los clasificadores débiles implementados, lo que indica que es más eficaz usar el Pipeline, a los clasificadores débiles por separado, sin mencionar el aumento del costo en tiempo por su puesta en marcha e implementación.

4. Conclusiones y recomendaciones

Conclusiones

- Se evidenció en el experimento de la sección **Ensamble *de novo,*** que la variación de conjuntos de *k-mer* para los ensambles *de novo*, para la métrica "errores de ensamble" generada por Quast V5.0.2, un efecto de las calidades de los procesos de secuenciación, pues la cantidad de errores encontrados en los ensambles era inversamente proporcional a la calidad de las lecturas, métrica calculada con FastQC, tampoco se encontró una relación con la combinación especifica de *k-mer*s de ensamble, como si se evidencio para las métricas de "Tamaño de ensamble" y "NG50". Cuyos mejores resultados se encontraron en una configuración de *k-mer*s distribuida a lo largo del espacio de búsqueda (21, 33, 55, 77, 99, 121), siendo también el conjunto de *k-mers* usados por *defaut* por la herramienta Spades.

- La optimización de los parámetros de selección de *contigs*, descrita en la sección **Ensamble de plásmidos con PlasmidSpades,** demostró como la herramienta de clasificación, es sensible a las zonas de baja complejidad del cromosoma, ya que resultan clasificados como falsos positivos, por la cantidad de conexiones en los grafos de ensamble, esto causa que su desempeño de clasificación se reduzca, siendo aun así, la segunda herramienta con mejor desempeño de los clasificadores débiles evaluados en la sección del **Multi-Clasificador** después de Similitud con Bases de Datos de Plásmidos.

- La comparación de los mejores conjuntos de clasificadores, enmarcado en los distintos tipos de algoritmo de multi-clasificador evaluados en la sección **Multi-clasificador,** demostró que la inclusión de metodologías con bajos desempeños como lo fueron profundidad de nucleótido y similitud contra bases de datos específicas de cromosoma**,** solo fueron de utilidad para el algoritmo de máquinas de soporte vectorial (SVM) con metodología *Staking*, superando también a las

demás metodologías evaluadas. Además, se identificó a la metodología *Bagging*, como la única que no mejoro el desempeño del mejor de sus clasificadores débiles.

- La evaluación mostrada en la sección **Evaluación Pipeline en especie ajena al entrenamiento,** permitió identificar de manera objetiva que las secuencias de cromosoma y de plásmidos comparten regiones, es decir hay una intersección de sus características, lo que dificulta con datos provenientes de secuenciación de nueva generación, su clasificación *INSILICO.*

Recomendaciones

La tarea de clasificación de secuencias de plásmidos desde el genoma completo *in silico*, es afectada en gran medida por la calidad de los datos de secuenciación de nueva generación, pues los tamaños a los que se termina fragmentando el genoma hacen del problema de ensamble *de novo* un tarea más compleja, es importante entonces pensar en la implementación de además de un método de secuenciación con bajo error como Ilumina, la inclusión de métodos de secuenciación de *reads* largos como *Oxford nano pore o Pasific Biosciences* y con metodologías de refinamiento o gapfilling, llenar aquellos huecos que fraccionan los genomas. Ya que la eficiencia de cualquiera de los métodos de clasificación evaluados en este trabajo, serian beneficiados por *contigs* más largos, por ende, elementos con mayor cantidad de características que faciliten su identificación y correcta clasificación.

El conocimiento sobre el comportamiento de la resistencia bacteriana continúa progresando, es importante para mantener la salida de este pipeline actualizada modificar las bases datos constantemente, pues la metodología de búsqueda por similitud es de gran relevancia para las clasificaciones realizadas por esta herramienta.

A. Anexo, Set de Datos recopilados

Set de datos de evaluación de herramientas de clasificación:
Link repositorio de GitHub

https://github.com/dctopro/Pipe_Plasmid/blob/main/Anexo_A.xlsx%20-%20Conjunto_Klebsiella.pdf

Nodos verdaderos

https://github.com/dctopro/Pipe_Plasmid/blob/main/tabla_nodos3.txt

B. Anexo, Evaluación de calidad de secuenciaciones adquiridas

Tabla con los datos referentes al proceso de evaluación y filtrado de las lecturas de secuenciación, del NCBI para la generación de un set de datos en condiciones normalisadas.

https://github.com/dctopro/Pipe_Plasmid/blob/main/set.xlsx%20-%20fastqc.pdf

C. Anexo, Evaluación de Ensambles en función del set de kmers

Tabla con los datos recopilados en el experimento de evaluación de conjunto óptimo de *k-mer*s, para la optimización de los ensambles *de novo*

https://github.com/dctopro/Pipe_Plasmid/blob/main/tabla_ensamble_ultima_version.zip

D. Anexo Experimento Evaluación de representatividad de *Contigs* PlasmidSpades desde Spades en función de las métricas de selección de alineamientos Blastn

Datos relacionados al experimento de elección de límites de desición en alineamientos entre los componentes generados por PlasmidSpades y los *contigs* de Spades.

https://github.com/dctopro/Pipe_Plasmid/blob/main/set.zip

E. Anexo Valores de *F-Score* de combinaciones especificas de

parámetros de selección Versus muestras del set de datos.

https://github.com/dctopro/Pipe_Plasmid/blob/main/tabla_fscore.zip

F. Anexo Exploración y optimización del método de clasificación por profundidad de nucleótido.

Análisis y optimización de límites de profundidad, para la clasificación de *contigs* de plásmidos.

https://github.com/dctopro/Pipe_Plasmid/blob/main/exploracion_profundidad.ipynb

https://github.com/dctopro/Pipe_Plasmid/blob/main/Maximizar_prof.ipynb

G. Anexo Optimización de método de selección por similitud y no-similares contra bases de datos.

https://github.com/dctopro/Pipe_Plasmid/blob/main/maximizacion_homologia_plasmidos.ipynb

https://github.com/dctopro/Pipe_Plasmid/blob/main/maximizacion_no_cro.ipynb

H. Optimización del set y modelo de unificación de resultados.

I. https://github.com/dctopro/Pipe_Plasmid/blob/main/Multi_modelo_5m.ipynb

J. Evaluación del PipeLine con una especie de bacteria ajena al entrenamiento.

https://github.com/dctopro/Pipe_Plasmid/blob/main/mm_pseudomonas.ipynb

k. Pipeline script, ejecución en segundo plano y manual de usuario

https://github.com/dctopro/Pipe_Plasmid/tree/main/pipeline

l. Evaluación del Pipe con set de datos y especie ajena al entrenamiento

https://github.com/dctopro/Pipe_Plasmid/blob/main/experimentos.py

Bibliografía

Aguilar-Bultet, L., & Falquet, L. (2015). Secuenciación y Ensamble de novo de genomas bacterianos: una alternativa para el estudio de nuevos patógenos. *Revista de Salud Animal*, *37*(2), 125–132.

Alimolaei, M., & Golchin, M. (2017). A comparison of methods for extracting plasmids from a difficult to lyse bacterium: Lactobacillus casei. *Biologicals*, *45*, 47–51. https://doi.org/10.1016/j.biologicals.2016.10.001

Antipov, D., Hartwick, N., Shen, M., Raiko, M., Lapidus, A., & Pevzner, P. A. (2016a). PlasmidSPAdes: Assembling plasmids from whole genome sequencing data. *Bioinformatics*, *32*(22), 3380–3387. https://doi.org/10.1093/bioinformatics/btw493

Antipov, D., Hartwick, N., Shen, M., Raiko, M., Lapidus, A., & Pevzner, P. A. (2016b). PlasmidSPAdes: Assembling plasmids from whole genome sequencing data. *Bioinformatics*, *32*(22), 3380–3387. https://doi.org/10.1093/bioinformatics/btw493

Antipov, D., Korobeynikov, A., McLean, J. S., & Pevzner, P. A. (2016). HybridSPAdes: An algorithm for hybrid assembly of short and long *reads*. *Bioinformatics*, *32*(7), 1009–1015. https://doi.org/10.1093/bioinformatics/btv688

Argemi, X., Martin, V., Loux, V., Dahyot, S., Lebeurre, J., Guffroy, A., … Prevost, G. (2017). Whole-Genome Sequencing of Seven Strains of Staphylococcus lugdunensis Allows Identification of Mobile Genetic Elements. *Genome Biology and Evolution*, *9*(5), 1183–1189. https://doi.org/10.1093/gbe/evx077

Benchmarking of de novo assembly tools: SPAdes 3.9 vs Velvet 1.2. (n.d.). Retrieved from https://cge.cbs.dtu.dk/services/cge/

Blair, J. M. A., Webber, M. A., Baylay, A. J., Ogbolu, D. O., & Piddock, L. J. V. (2015). Molecular mechanisms of antibiotic resistance. *Nature Reviews Microbiology*, *13*(1), 42–51. https://doi.org/10.1038/nrmicro3380

Bootsma, H. J., & Schouls, L. M. (2015, March 1). Next-generation sequencing of carbapenem-resistant Gram-negative microorganisms: A key tool for surveillance and infection control. *Future Microbiology*, *10*(3), 299–302.

https://doi.org/10.2217/FMB.14.134

Bousquet, A., Henquet, S., Compain, F., Genel, N., Arlet, G., & Decré, D. (2015). SC. *Journal of Microbiological Methods*. https://doi.org/10.1016/j.mimet.2015.01.019

Bryson, K., Loux, V., Bossy, R., Nicolas, P., Chaillou, S., Guchte, M. Van De, … Lactique, F. (2006). *AGMIAL : implementing an annotation strategy for prokaryote genomes as a distributed system*. *34*(12), 3533–3545. https://doi.org/10.1093/nar/gkl471

C., L., N., G., S., L., C., G., & G., M. M. (2017). Multi-clasificador para predecir interacción de proteínas usando optimización basada en colonia de hormigas. *Revista Cubana de Ciencias Informáticas*, *11*, 195–210. Retrieved from https://www.redalyc.org/articulo.oa?id=378349711014

Cabrera-Hernández, L., Morales-Hernández, A., & Casas-Cardoso, G. M. (2016). Medidas de diversidad para la construcción de sistemas multi-clasificadores usando algoritmos genéticos. *Computacion y Sistemas*, *20*(4), 729–747. https://doi.org/10.13053/CyS-20-4-2513

Carattoli, A., Zankari, E., Garciá-Fernández, A., Larsen, M. V., Lund, O., Villa, L., … Hasman, H. (2014). In Silico detection and typing of plasmids using plasmidfinder and plasmid multilocus sequence typing. *Antimicrobial Agents and Chemotherapy*, *58*(7), 3895–3903. https://doi.org/10.1128/AAC.02412-14

Chaisson, M., Pevzner, P., & Tang, H. (2004). Fragment assembly with short *reads*. *Bioinformatics*, *20*(13), 2067–2074. https://doi.org/10.1093/bioinformatics/bth205

Chikhi, R., & Medvedev, P. (2014). Informed and automated *k-mer* size selection for genome assembly. *Bioinformatics*, *30*(1), 31–37. https://doi.org/10.1093/bioinformatics/btt310

Durai, D. A., & Schulz, M. H. (2016). Informed kmer selection for de novo transcriptome assembly. *Bioinformatics*, *32*(11), 1670–1677. https://doi.org/10.1093/bioinformatics/btw217

Fang, Z. C., Tan, J., Wu, S. F., Li, M., Xu, C. M., Xie, Z. J., & Zhu, H. Q. (2019). PPR-Meta: a tool for identifying phages and plasmids from metagenomic fragments using deep learning. *Gigascience*, *8*(6). https://doi.org/10.1093/gigascience/giz066

Founou, R. C., Founou, L. L., & Essack, S. Y. (2018). Extended spectrum beta-lactamase mediated resistance in carriage and clinical gram-negative ESKAPE bacteria: a comparative study between a district and tertiary hospital in South Africa. *Antimicrobial Resistance and Infection Control*, *7*. https://doi.org/10.1186/s13756-018-0423-0

Kim, D., Song, L., Breitwieser, F. P., & Salzberg, S. L. (2016). Centrifuge: Rapid and

sensitive classification of metagenomic sequences. *Genome Research*, *26*(12), 1721–1729. https://doi.org/10.1101/gr.210641.116

Kotsianti, S. B., & Kanellopoulos, D. (2007). Combining *Bagging*, *Boosting* and Dagging for Classification Problems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4693 LNAI*(PART 2), 493–500. https://doi.org/10.1007/978-3-540-74827-4_62

Krawczyk, P S, Lipinski, L., & Dziembowski, A. (2018a). PlasFlow: predicting plasmid sequences in metagenomic data using genome signatures. *Nucleic Acids Research*, *46*(6). https://doi.org/10.1093/nar/gkx1321

Krawczyk, P S, Lipinski, L., & Dziembowski, A. (2018b). PlasFlow: predicting plasmid sequences in metagenomic data using genome signatures. *Nucleic Acids Research*, *46*(6). https://doi.org/10.1093/nar/gkx1321

Krawczyk, Pawel S, Lipinski, L., & Dziembowski, A. (2018). PlasFlow: predicting plasmid sequences in metagenomic data using genome signatures. *Nucleic Acids Research*, *46*(6), e35–e35. https://doi.org/10.1093/nar/gkx1321

Le Roux, C., Huet, G., Jauneau, A., Camborde, L., Tremousaygue, D., Kraut, A., … Deslandes, L. (2015). A Receptor Pair with an Integrated Decoy Converts Pathogen Disabling of Transcription Factors to Immunity. *Cell*, *161*(5), 1074–1088. https://doi.org/10.1016/j.cell.2015.04.025

Li, D., Liu, C. M., Luo, R., Sadakane, K., & Lam, T. W. (2015). MEGAHIT: An ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, *31*(10), 1674–1676. https://doi.org/10.1093/bioinformatics/btv033

Li, X. Z., Plesiat, P., & Nikaido, H. (2015). The Challenge of Efflux-Mediated Antibiotic Resistance in Gram-Negative Bacteria. *Clinical Microbiology Reviews*, *28*(2), 337–418. https://doi.org/10.1128/cmr.00117-14

Liu, Y. Y., Wang, Y., Walsh, T. R., Yi, L. X., Zhang, R., Spencer, J., … Shen, J. Z. (2016). Emergence of plasmid-mediated colistin resistance mechanism MCR-1 in animals and human beings in China: a microbiological and molecular biological study. *Lancet Infectious Diseases*, *16*(2), 161–168. https://doi.org/10.1016/s1473-3099(15)00424-7

Lowe, T. M., & Eddy, S. R. (1997). tRNAscan-SE: A Program for Improved Detection of Transfer RNA Genes in Genomic Sequence. *Nucleic Acids Research*, *25*(5), 955–964. https://doi.org/10.1093/nar/25.5.955

Marçais, G., & Kingsford, C. (2011). A fast, lock-free approach for efficient parallel counting of occurrences of *k-mer*s. *Bioinformatics*, *27*(6), 764–770. https://doi.org/10.1093/bioinformatics/btr011

Martinez, J. L., Coque, T. M., & Baquero, F. (2015). What is a resistance gene? Ranking risk in resistomes. *Nature Reviews Microbiology*, *13*(2), 116–123. https://doi.org/10.1038/nrmicro3399

Nishida, H. (2012). Comparative Analyses of Base Compositions, DNA Sizes, and Dinucleotide Frequency Profiles in Archaeal and Bacterial Chromosomes and Plasmids. *International Journal of Evolutionary Biology*, *2012*, 1–5. https://doi.org/10.1155/2012/342482

Olsen, I. (2015). Biofilm-specific antibiotic tolerance and resistance. *European Journal of Clinical Microbiology & Infectious Diseases*, *34*(5), 877–886. https://doi.org/10.1007/s10096-015-2323-z

Otzen, T., & Manterola, C. (2017). Técnicas de Muestreo sobre una Población a Estudio. *International Journal of Morphology*, *35*(1), 227–232. https://doi.org/10.4067/S0717-95022017000100037

Program, B., & Guide, S. (2009). *BLAST Basic Local Alignment Search Tool*. 1–20.

Ren, J., Ahlgren, N. A., Lu, Y. Y., Fuhrman, J. A., & Sun, F. (2017). VirFinder: a novel *k-mer* based tool for identifying viral sequences from assembled metagenomic data. *Microbiome*, *5*(1), 69. https://doi.org/10.1186/s40168-017-0283-5

Ross, D. L., & Schultz, R. K. (1996). Effect of inhalation flow rate on the dosing characteristics of dry powder inhaler (DPI) and metered dose inhaler (MDI) products. *Journal of Aerosol Medicine: Deposition, Clearance, and Effects in the Lung*, *9*(2), 215–226. https://doi.org/10.1089/jam.1996.9.215

Royer, G., Decousser, J. W., Branger, C., Dubois, M., Médigue, C., Denamur, E., & Vallenet, D. (2018). PlaScope: a targeted approach to assess the plasmidome from genome assemblies at the species level. *Microbial Genomics*, *4*(9), 1–8. https://doi.org/10.1099/mgen.0.000211

Rubio, S., Pacheco-Orozco, R. A., Gómez, A. M., Perdomo, S., & García-Robles, R. (2020). Secuenciación de nueva generación (NGS) de ADN: presente y futuro en la práctica clínica. *Universitas Médica*, *61*(2). https://doi.org/10.11144/javeriana.umed61-2.sngs

Schroeder, M., Brooks, B. D., & Brooks, A. E. (2017, January 18). The complex relationship between virulence and antibiotic resistance. *Genes*, Vol. 8. https://doi.org/10.3390/genes8010039

Torres Manno, M. A., Pizarro, M. D., Prunello, M., Magni, C., Daurelio, L. D., & Espariz, M. (2019). GeM-Pro: a tool for genome functional mining and microbial profiling. *Applied Microbiology and Biotechnology*, *103*(7), 3123–3134. https://doi.org/10.1007/s00253-019-09648-8

Van Horn, C., Wu, F. N., Zheng, Z., Dai, Z. H., & Chen, J. C. (2019). Detection of a Single-Copy Plasmid, pXFSL21, in Xylella fastidiosa Strain Stag's Leap with Two Toxin-Antitoxin Systems Using Next-Generation Sequencing. *Phytopathology*, *109*(2), 240–247. https://doi.org/10.1094/phyto-07-18-0249-fi

WHO. (2017). Global priority list of antibiotic-resistant batceria to guide research, discovery, and development of new antibiotics. *Who*, 7. Retrieved from https://www.who.int/news-room/detail/27-02-2017-who-publishes-list-of-bacteria-for-which-new-antibiotics-are-urgently-needed

Wootton, J. C. (1994). Non-globular domains in protein sequences: Automated segmentation using complexity measures. *Computers and Chemistry*, *18*(3), 269–285. https://doi.org/10.1016/0097-8485(94)85023-2

Xavier, B. B., Sabirova, J., Pieter, M., Hernalsteens, J. P., De Greve, H., Goossens, H., & Malhotra-Kumar, S. (2014). Employing whole genome mapping for optimal de novo assembly of bacterial genomes. *BMC Research Notes*, *7*(1), 1–4. https://doi.org/10.1186/1756-0500-7-484

Zhou, F., & Xu, Y. (2010). cBar: A computer program to distinguish plasmid-derived from chromosome-derived sequence fragments in metagenomics data. *Bioinformatics*, *26*(16), 2051–2052. https://doi.org/10.1093/bioinformatics/btq299