UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA AGRONÓMICA Y DEL MEDIO NATURAL

GRADO EN BIOTECNOLOGÍA

CREACIÓN DE UN PIPELINE PARA EL ANÁLISIS DE DATOS METAGENÓMICOS BASADO EN NEXTFLOW

TRABAJO FIN DE GRADO

Curso Académico: 2020/2021
Valencia, Julio de 2021

Alumna: Alba Lomas Redondo
Tutor: José Javier Forment Millet
Segundo Tutor: Giuseppe D'Auria

TÍTULO: Creación de un pipeline para el análisis de datos metagenómicos basado en Nextflow.

RESUMEN

La metagenómica es una disciplina que surge ante la imposibilidad de aislar y cultivar la (amplia) mayoría de los organismos microbianos que viven en los ecosistemas naturales. El término hace referencia al estudio del conjunto de genomas de los organismos que habitan una muestra natural, al fin de describir, quiénes son y cuáles son sus potencialidades genéticas. Gracias a los considerables avances en el campo de la biología molecular, las nuevas tecnologías de secuenciación masiva (Next Generation Sequencing, NGS) y la bioinformática, la metagenómica se postula como una de las disciplinas con mayor progreso, en la actualidad, en el campo de la ecología microbiana.

Una parte fundamental del análisis metagenómico son las herramientas bioinformáticas empleadas para analizar los resultados de secuenciación; aunque esta disciplina puede considerarse bastante reciente, existen ya una gran variedad de bases de datos, *pipelines* (flujo de trabajo) y programas. Con respecto a los pipelines, estos se desarrollan para la integración de diversos paquetes de software complementarios y para la automatización de los procesos; no obstante, al aumentar la complejidad del análisis aparecen obstáculos que dificultan el uso de pipelines de alto rendimiento: incompatibilidad entre los diferentes paquetes de software, requisitos de actualización contradictorios, gestión de un elevado número de archivos intermedios y temporales u optimización de los recursos de computación. A fin de solventar todos estos problemas, recientemente ha surgido un lenguaje específico de dominio (DLS) llamado Nextflow. Este permite la adaptación de pipelines escritos en cualquier lenguaje de programación. La elección de Nextflow para el desarrollo de un pipeline para análisis metagenómicos se justifica por características como el uso de tecnologías de contenedores multi-escala, su integración en repositorios de software, la paralelización y la definición de canales de entrada y salida para el inicio automático de cada sub-proceso. Además, el modelo de flujo de datos mejora a otras herramientas alternativas; ya que, el procesamiento “top-down” no necesita gran espacio de almacenamiento. En definitiva, Nextflow se presenta como una solución flexible y robusta debido a la simplificación, al control del flujo de datos y a la gestión de los resultados que se recogen de los análisis metagenómicos.

En esta tesis, se presenta un pipeline de análisis basado en Nextflow que recoge los pasos principales del análisis metagenómico. El pipeline propuesto proporciona al usuario una herramienta que evita el proceso de instalación de los programas necesarios en el análisis y ejecuta los pasos más dispendiosos desde el punto de vista computacional en el análisis metagenómico. El pipeline recibe como entrada los datos brutos a analizar, procedentes de una o más muestras y continua con los pasos de control de calidad, ensamblado, anotación y cuantificación de cada anotación. Finalmente, proporciona, de forma clara y resumida, los resultados necesarios para los siguientes análisis estadísticos descriptivos y/o diferenciales; actuando de forma transparente, robusta y altamente reproducible, acelerando los tiempos de ejecución, ahorrando espacio de almacenamiento para el análisis y optimizando así los recursos informáticos disponibles.

PALABRAS CLAVE: metagenómica, bioinformática, pipeline, Nextflow.

AUTORA: Dña. Alba Lomas Redondo
TUTOR ACADÉMICO: Prof. D. José Javier Forment Millet
SEGUNDO TUTOR: Prof. D. Giuseppe D’Auria

Valencia, julio 2021

TITLE: Creation of a Nextflow based pipeline for metagenomic data analysis.

ABSTRACT
Metagenomics is a discipline that arose from the impossibility of isolating and cultivating most of microbial organisms that live in natural ecosystems. The term refers to the study of the genomes of the organisms inhabiting a natural environment in order to describe which they are and their genetic potentialities. Because of the considerable advances in the field of molecular biology, Next Generation Sequencing technologies (NGS) and bioinformatics, metagenomics is currently one of the most challenging disciplines in the field of microbial ecology.

The analysis of sequencing results is performed by bioinformatic tools that are an essential part of metagenomic analysis; although this discipline may be considered quite recent, a wide variety of databases, pipelines (workflows) and programs have been developed. Pipelines are used for the integration of different and complementary software packages, and for the automation of processes. However, as the complexity of the analysis increases, obstacles which hinder the use of high-performance pipelines appear incompatibility between different software packages, conflicting upgrade requirements, management of a large number of intermediate and temporary files, and optimization of computing resources. In order to overcome all these problems, a domain-specific language (DLS) called Nextflow has recently emerged. Nextflow allows the adaptation of pipelines written in any programming language. The choice of Nextflow for the development of a metagenomic analysis pipeline is justified by features such as the use of multi-scale containerization, its integration with software repositories, parallelization and the definition of input and output channels for the automatic start of each sub-process. In addition, the data flow model improves other alternative tools, because the top to bottom processing does not require large storage space. In conclusion, Nextflow stands to be a flexible and robust solution owing to the simplification, data flow control and management of the results collected from metagenomic analysis.

In this thesis, a Nextflow-based analysis pipeline, that captures the main steps of metagenomic analysis, is presented. The workflow provides the user with a tool that avoids the process of installing the necessary programs for the analysis and executes the most complex and computationally problematic stages in metagenomic analysis. The pipeline receives as input the raw data to be analyzed from one or more samples and it continues with the steps of quality control, assembly, annotation, and quantification of each annotation. Finally, it provides, in a clear and summarized way, the necessary results for the following descriptive and/or differential statistical analysis, acting in a transparent, robust, and highly reproducible way, speeding up execution times, saving storage space for the analysis, and optimizing the available informatics resources.

KEY WORDS: metagenomics, bioinformatics, pipeline, Nextflow.
AUTHOR: Dña. Alba Lomas Redondo
ACADEMIC TUTOR: Prof. D. José Javier Forment Millet
SECOND TUTOR: Prof. D. Giuseppe D'Auria

Valencia, July 2021

HOJA DE AGRADECIMIENTOS

A Giuseppe, por acogerme y ofrecerme su tiempo. Su ayuda ha sido imprescindible y su paciencia, infinita.

A Javier por aceptar ser mi tutor y responder todas las dudas que le he planteado instantáneamente.

A mi familia, gracias a ellos he podido llegar hasta aquí, gracias por creer en mí. Especialmente a María que ha lidiado con mis cuatro años de carrera y veintiuno de vida.

A todos los que me han acompañado durante estos cuatro años, sin vosotros no hubiera sido posible.

ÍNDICE

1. INTRODUCCIÓN 1
1.1. TÉCNICAS DE SECUENCIACIÓN 2
1.2. DEFINICIÓN DE PIPELINE 4
1.3. PIPELINES EN METAGENÓMICA 5
2. PROCESOS BÁSICOS EN UN ANÁLISIS METAGENÓMICO 7
2.1. CONTROL DE CALIDAD 7
2.2. ENSAMBLADO 8
2.3. ANOTACIÓN 9
2.3.1. Anotación estructural 11
2.3.2. Anotación funcional 12
2.3.3. Bases de datos 13
2.4. MAPEO 13
2.5. EXPRESIÓN DIFERENCIAL 15
3. RESULTADOS 16
3.1. NEXTFLOW (DSL1) 16
3.2. CONSIDERACIONES INICIALES 18
3.3. DEFINICIÓN DE PARÁMETROS 18
3.4. PROCESO LIMPIA 19
3.5. PROCESO CONCAT1 y CONCAT2 20
3.6. PROCESO ASSEMBLY 21
3.7. PROCESO PROKKA 22
3.8. PROCESO annotationKegg 22
3.9. PROCESO annotationCog 23
3.10. PROCESO annotationPfam 24
3.11. PROCESO MAPPING 24
3.12. PROCESO HTSEQ 25
4. BENCHMARKING 27
6. BIBLIOGRAFÍA 29
7. ANEXO I 38
8. ANEXO II 42

ÍNDICE DE TABLAS

- **Tabla 1:** Características de diferentes pipelines de análisis metagenómico, en comparación con SqueezeMeta (Tamames y Puente-Sánchez, 2018).

- **Tabla 2:** Herramientas de predicción de características utilizadas por Prokka (Seemann, 2014).

- **Tabla 3:** Descripción de los archivos de salida de Prokka (Seemann, 2014).

- **Tabla 4:** Comparación entre Nextflow y distintas herramientas de gestión de flujos (Di Tomaso *et al.*, 2017).

- **Tabla 5:** Descripción de los campos obligatorios que aparecen en un archivo con formato SAM (The SAM/BAM Format Specification Working Group, 2021).

ÍNDICE DE FIGURAS

- **Figura 1:** Visión general de las tecnologías de secuenciación de segunda y tercera generación (Ambardar *et al*., 2016).

- **Figura 2**: Estrategias de tecnologías de tercera generación (TGS) (Schadt *et al.*, 2010)**.**

- **Figura 3:** Esquema de los estados de un pipeline genérico (Veloso, 2018).

- **Figura 4:** Esquema comparativo del funcionamiento de orden FIFO y LIFO.

- **Figura 5:** Estructura básica de dependencia de un modelo oculto de Márkov (HMM) con una secuencia observada que surge de una secuencia no observada (McClintock *et al*.; 2020).

- **Figura 6:** Esquema de *main.nf.*

- **Figura 7:** A) Gráfico del uso de la CPU en cada proceso en *main.nf*. B) Gráfico del uso de la memoria RAM de cada proceso en *main.nf*.

- **Figura 8:** Cronología de la ejecución de los procesos de *main.nf.*

- **Figura 9:** Ejemplo visual de un archivo formato FASTA (NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION [NCBI], s.f.).

- **Figura 10:** Ejemplo visual de un archivo formato FASTQ (ILLUMINA, 2020a).

- **Figura 11:** Archivo "diamond.ylm" que contiene las instrucciones para crear el ambiente conda que alberga el paquete de software DIAMOND (0.9.22).

- **Figura 12:** Archivo "prokka.ylm" que contiene las instrucciones para crear el ambiente conda que alberga el paquete de software PROKKA (1.14.6).

- **Figura 13:** Ejemplo visual de un archivo formato SAM (THE SAM/BAM FORMAT SPECIFICATION WORKING GROUP, 2021).

- **Figura 14:** Ejemplo visual de un archivo formato GTF (ENSEMBL, s.f.).

- **Figura 15:** Ejemplo visual de un archivo formato GFF (ENSEMBL, s.f.).

- **Figura 16:** Esquema de la estrategia mate pair (ECSEQ BIOINFORMATICS, 2017).

- **Figura 17:** Esquema de la estrategia *paired end* (ECSEQ BIOINFORMATICS, 2017).

- **Figura 18:** Ejemplo visual del grafo de Bruijn (Compeau, P. *et al*.; 2011).

- **Figura 19:** Comparación entre un grafo de Bruijn (A) y un grafo emparejado de Bruijn (B) (Medvedev *et al*., 2011).

- **Figura 20:** Representación de un DAG, los nodos se han nombrado aleatoriamente.

1. INTRODUCCIÓN

La metagenómica es una rama de la genómica, que nace hacia finales del siglo XX, fue en el 1998 cuando fue citada por primera vez en un artículo científico (Handelsman *et al*., 1998).

Esta disciplina surge ante la necesidad de sortear un problema acuciante en el campo de la microbiología provocado por la dificultad que suponía – y supone – el poder cultivar y conservar la mayoría de lo organismos microbianos que viven en condiciones y ecosistemas naturales. La imposibilidad de cultivo se traduce a su vez en la imposibilidad de acceder a la información genómica de estos microorganismos.

En una charla en la Estación Experimental del Zaidín (EEZ), centro perteneciente al Consejo Superior de Investigaciones Científicas (CSIC) en 2010, el profesor Francisco Rodríguez Valera, investigador del Grupo de Genómica Evolutiva de la Universidad Miguel Hernández (San Juan de Alicante), afirmaba que "Desde que los grandes pioneros de esta ciencia sentaran las bases del cultivo puro nunca un cambio tecnológico había abierto tantas posibilidades. Por primera vez los investigadores pueden profundizar en el conocimiento del 99% de los microbios que son difícilmente cultivables".

El término "metagenómica" está conformado por dos palabras: "meta" que se interpreta como "más allá" y "genómica", un campo de la biología molecular relacionado con el estudio de los genomas (NATIONAL HUMAN GENOME RESEARCH INSTITUTE [NHGRI], 2021). Es decir, se habla de metagenómica cuando lo que se está analizando es una muestra en la que vive una comunidad microbiana constituida por diferentes microorganismos que se pueden estudiar solo de forma conjunta. Para entender el potencial de la metagenómica basta con pensar en los estudios que describen el microbioma gastrointestinal humano o los microbiomas de otros ambientes como el lecho marino, la rizosfera, los ambientes extremos salinos, árticos, etc. En estos casos, resulta muy difícil o imposible centrar el foco del estudio en un solo un tipo de microorganismo, perteneciente a una comunidad. De este modo, lo que se pretende es inferir información funcional y taxonómica de una muestra compleja sin realizar un cultivo previo (Handelsman *et al*., 2002).

La metagenómica debe su desarrollo a la evolución de las técnicas de biología molecular, métodos de secuenciación de segunda, y ahora tercera, generación (Next Generation Sequencing: NGS, representadas en la Figura 1) y la bioinformática, entre otros.

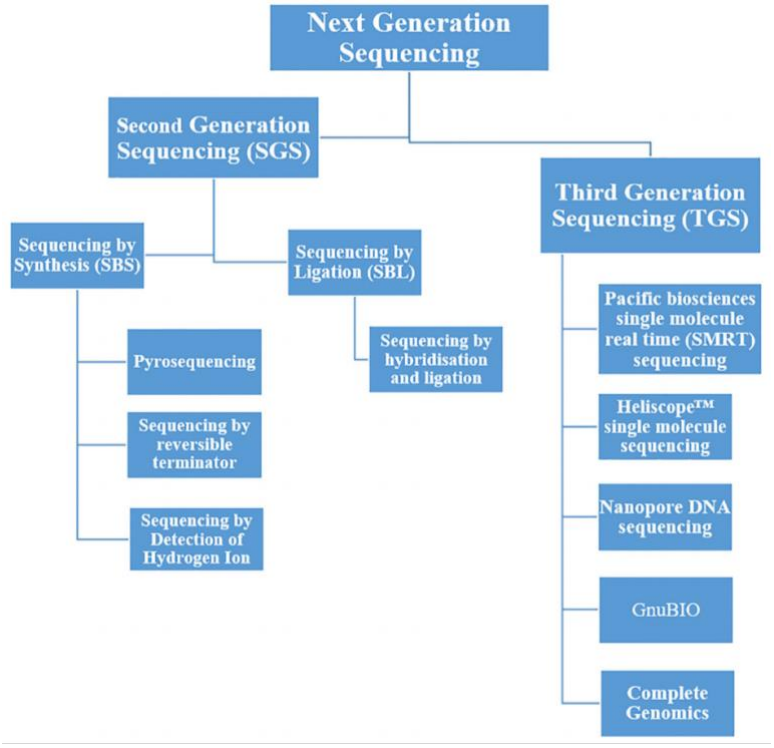

Figura 1. Visión general de las tecnologías de secuenciación de segunda y tercera generación (Ambardar *et al*., 2016).

1.1. TÉCNICAS DE SECUENCIACIÓN

En lo referente a las técnicas de secuenciación, ha habido un desarrollo vertiginoso de estas que ha dado como fruto diversas generaciones de técnicas de secuenciación desde 1977 hasta hoy. En concreto existen tres generaciones; la primera de ellas tiene como máximo exponente la secuenciación tipo Sanger o método de terminación de cadena (Sanger y Coulson, 1975 y Sanger *et al*., 1977). Las siguientes generaciones (NGS) aparecieron hacia 2004 con 454 Roche GS FLX System, ya en desuso, donde, por primera vez se pude secuenciar de manera paralela cientos de miles de fragmentos de DNA en un solo experimento con fragmentos que no superan los 300 pares de bases (bp) (segunda generación, Voelkerding *et al*., 2009). Las técnicas de secuenciación de tercera generación, cuyas estrategias se resumen en la Figura 2, emergieron en 2010, permitiendo secuenciar fragmentos de hasta cientos de miles de pares de bases como la tecnología de Pacific Biosciences *single-molecule real*-time (SMRT) *sequencing* (ver Anexo II) (Rhoads y Au, 2015) o las basadas en nanoporos de Oxford Nanopore Technologies (Lu *et al*., 2016).

En un primer lugar, se desarrollaron dos técnicas de primera generación, la que más impacto tuvo -y tiene- es la secuenciación Sanger que se basó en la electroforesis capilar individual de los productos derivados de la reacción de secuenciación marcados con fluorescencia (Mardis, 2011); la otra, que surgió de la mano de Maxam y Gilbert se denominó método de secuenciación química (Maxam y Gilbert, 1977). No obstante, este último fue totalmente eclipsado por el primero debido a su complejidad. Por otro lado, las NGS pretendían suplir algunas de las desventajas de la secuenciación tipo Sanger, como la lentitud del proceso y el coste, a través de la paralelización de la secuenciación y la introducción de la PCR (ver Anexo II) en el proceso. Existen dos tipos de técnicas dentro de esta generación: síntesis de moléculas de DNA (como ocurre en método de Sanger) y ligación de oligonucleótidos – SOLiD de Thermofisher –. Por último, las tecnologías de tercera generación también surgen de la necesidad de mejorar a sus antecesoras que presentan dos limitaciones: dificultad en el ensamblaje por la longitud insuficiente de las lecturas y el sesgo de la PCR introducido por la amplificación clonal (Ambardar *et al*., 2016 y Schadt *et al*., 2010). En general la estrategia de las técnicas de tercera generación pretende eliminar la amplificación por PCR, obtener lecturas más largas y secuenciar con una única molécula de DNA, *single-molecule secuencing* (SMS) (ver Anexo II). Estas se pueden clasificar, a grandes rasgos, en tres grupos: secuenciación por síntesis (SBS) (ver Anexo II), fue el primer enfoque de tercera generación y como resultado la secuenciación a tiempo real de molécula única (SMRT) de PacBio; secuenciación por nanoporos, ejemplificada por las técnicas de Oxford Nanopore Technologies (ONT) (Clarke *et al*., 2009) y el uso de técnicas avanzadas de microscopía para obtener imágenes directas de moléculas de DNA individuales (Schadt *et al*., 2010).

A parte de las técnicas ya nombradas existen otras plataformas (Ambardar *et al*., 2016):

- Segunda generación: Genome Analyzer, HiSeq, MiSeq y NextSeq de Illumina, Inc., e Ion Torrent de Life Technologies*.
- Tercera generación: LLC, HelicosTM Genetic Analysis System de SeqLL o Complete Genomics de Beijing Genomics Institute y GnuBIO de BioRad.

*Ion Torrent se sitúa a medio camino entre ambas generaciones, este es un secuenciador totalmente diferente a los demás; emplea una tecnología de semiconductores de última generación, pues el dispositivo es un chip de silicio grabado con una matriz de “pozos” nanoscópicos que descansan sobre una capa sensible a iones (Pennisi E., 2010.)

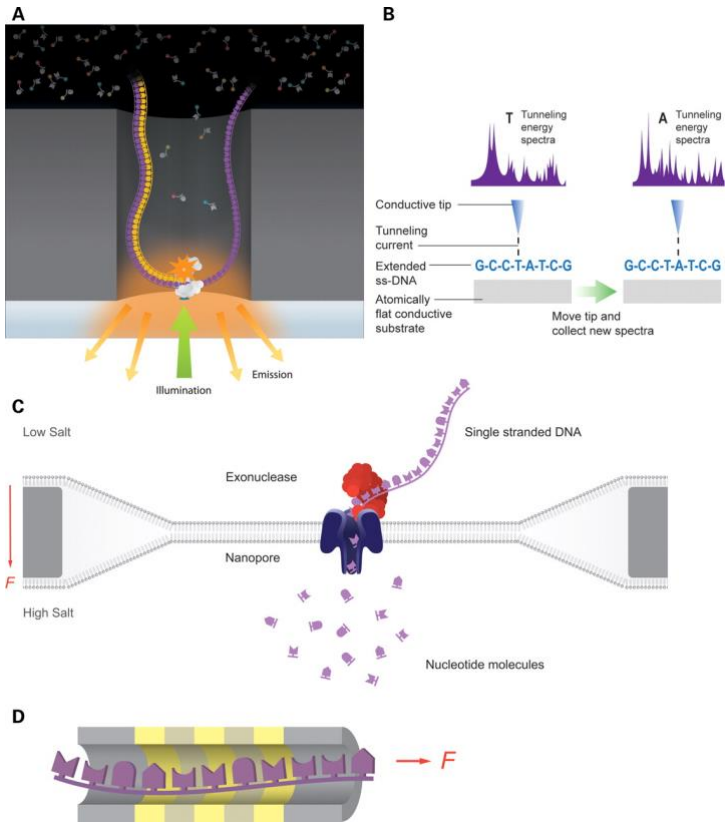

Figura 2. Estrategias de tecnologías de tercera generación (TGS). Las TGS son capaces de analizar moléculas únicas, prescindiendo de pasos como el lavado durante la síntesis de DNA. (**A**) Tecnología PacBio (SMRT), se inmoviliza una DNA polimerasa en una ZMW (ver Anexo II) y se miden las adiciones de bases por detección de fluorescencia de fosfonucleótidos marcados con rayos gamma. (**B**) Tecnología Reveo, se examina una molécula de ssDNA mediante STM (microscopio de Efeco Túnel). Existen varias compañías que secuencian con tecnología similar, empleando microscopía electrónica. (**C**) Tecnología Oxford Nanopore que secuencia a través de un poro, midiendo la translocación de nucleótidos escindidos, impulsados por la fuerza de las concentraciones diferenciales de iones a través de la membrana. (**D**) Tecnología de transistores de DNA de IBM, la secuenciación se produce mediante la lectura de bases individuales al pasar por una abertura estrecha, el reconocimiento de cada nucleótido se realiza por su firma electrónica única. Las bandas doradas representan las capas metálicas y las grises las dieléctricas del transistor (Schadt *et al*., 2010).

Con el crecimiento de la producción de datos de secuencias mediante la entrada de los nuevos métodos de secuenciación masiva e *in situ*, acoplado a la caída de los precios de secuenciación por nucleótido y los tiempos de obtención de datos, hoy en día la metagenómica es un proceso de fácil acceso con el potencial de producir un alto nivel de información genómica a un coste siempre mas accesible. En los últimos años, debido a la explosión de estos métodos, el fulcro de los análisis metagenómicos se ha desplazado desde los laboratorios para la obtención y procesamiento de las muestras hacía el análisis bioinformático. Lo cierto es que, una vez recibidos los datos de secuencias, el nuevo desafío se lleva a cabo en el territorio de la bioinformática.
Los procesos analíticos usados en metagenómica necesitan de herramientas bioinformáticas que lleven a cabo los pasos de control de calidad, reconstrucción de los genomas de los microorganismos presentes en las muestras, anotación filogenética y funcional, cuantificación y estadística.

Todos los procesos requieren de servidores adecuados, tiempos de cálculo y capacidad de almacenamiento de un cierto calibre. Por ello, uno de los desafíos en el análisis metagenómico, objetivo principal de esta tesis es: optimizar los procesos que requieren más recursos computacionales para poder analizar de forma ágil, rápida y robusta los datos de secuenciación de proyectos metagenómicos.

Para cubrir las necesidades del análisis han surgido una gran cantidad programas y flujos de trabajo, llamados en jerga “pipelines” o tuberías. “Los procesos básicos que deben llevar a cabo consisten en un control de calidad de las secuencias primarias, su ensamblaje, predicción de genes, y la anotación funcional y taxonómica de los genes identificados. La gran mayoría de los pipelines de análisis metagenómicos cubren estos pasos (Kim *et al*., 2016, Li y Durbin, 2009, Eren *et al*., 2015, Arumugam *et al*., 2010 y Abubucker *et al*., 2012,). Sin embargo, difieren en capacidad y enfoque” (Tamames y Puente-Sánchez, 2018).

1.2. DEFINICIÓN DE PIPELINE

En el ámbito informático, la arquitectura en pipeline comprende un conjunto de procesos o fases secuenciales por los que va circulando un flujo de datos para ser transformado. Como consecuencia, estos procesos están conectados entre sí y, normalmente, la salida de una determinada fase es la entrada de la siguiente. En otras palabras, un pipeline, engloba una cadena de diferentes procesos que “beben unos de otros”.
Dentro de la estructura de pipeline, se pueden encontrar diferentes estados por los que debe pasar cada proceso parte de la “tubería virtual”; estos involucran desde la carga de la instrucción a ejecutar y los datos iniciales, hasta la escritura y el almacenamiento de los datos que resultan tras la ejecución de la instrucción principal.
Se puede simplificar como un esquema que interpreta un flujo constante de información, procesándola de forma secuencial. La comunicación de los procesos o datos a través de los pipelines se basa en una interacción productor/consumidor: la información necesaria para el proceso consumidor viene dada por el resultado del proceso productor anterior; es importante resaltar que el almacenamiento de los datos intermedios entre un proceso y el siguiente es temporal.
Además, los pipelines tienen diversas características que los hacen muy comunes en sistemas operativos multitarea:

- capacidad de multiprogramación, empleando distintos lenguajes o llamadas a programas de distinta naturaleza;
- análisis en paralelo o secuencial según las necesidades y posibilidades;
- capacidad de ejecutar procesos independientes de manera simultánea.

La estructura general alberga diferentes estados por los que debe pasar cada proceso parte de la “tubería virtual”. Los estados involucran desde la carga de la instrucción a ejecutar y los datos iniciales, hasta la escritura y el almacenaje de los datos que resultan tras la ejecución de la instrucción principal, como muestra la Figura 3.

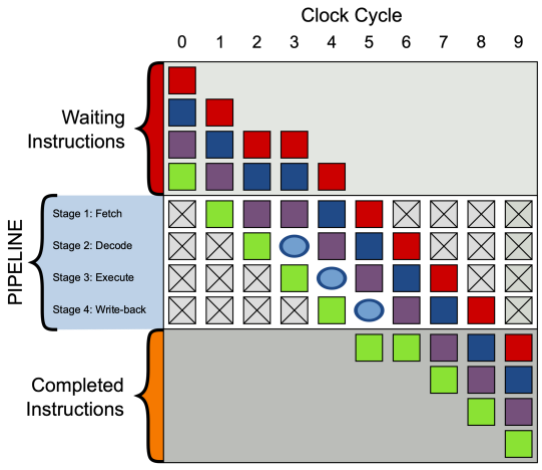

Figura 3. Esquema de los estados de un pipeline genérico. Estado 1. Recuperación (*Fetch*): se encarga de traer la instrucción que se debe ejecutar, de la dirección que está almacenada en el contador del programa.

Estado 2. Decodificación (*Decode*): se encarga de guardar o almacenar la instrucción en el registro de instrucciones y luego descifrarla. Estado 3. Ejecución (*Execute*): se ejecuta la instrucción almacenada en el registro de instrucciones. Estado 4. Escritura en retroceso (*Write-back*): "escribe" los resultados de esa instrucción dentro del registro de destinación (Veloso, 2018).

Como se ha explicado con anterioridad, dentro de un pipeline existe un proceso "*progenitor*" que alimenta a un proceso "*hijo*" a través de sus datos de salida que deben ser almacenados al menos hasta que el programa el proceso *hijo* los necesite. Por lo tanto, hay que tener presente un nuevo concepto: almacenamiento en buffer. Este tipo de memoria asegura que el consumidor obtenga las entradas necesarias para su correcto funcionamiento; incluso cuando la velocidad de transferencia de datos es irregular.
Por otra parte, la comunicación entre los diferentes procesos que conforman cualquier cauce sigue un orden FIFO (del inglés: *first in, first out*). El concepto está basado en la semejanza entre la trayectoria que sigue la información dentro de este tipo de proceso y el funcionamiento de las colas de atención al usuario: el que llega primero, es el que se marcha antes, queda representado gráficamente en la Figura 4.

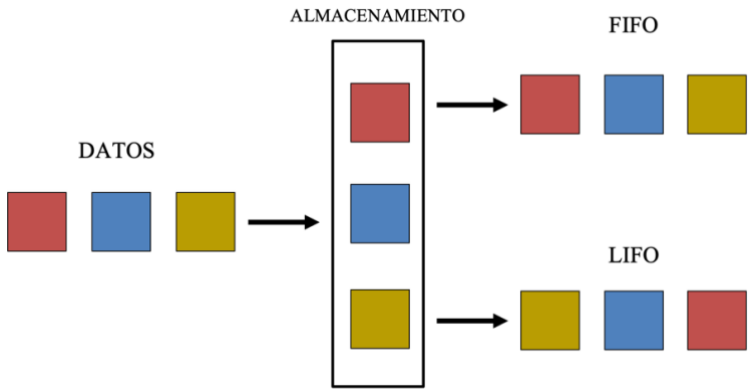

Figura 4. Esquema comparativo del funcionamiento de orden FIFO y LIFO.

Finalmente, podemos resumir el concepto de pipeline como un modelo informático capaz de segmentar un conjunto de procesos con la finalidad de poder ejecutarlos -gracias a un administrador de tareas- de manera secuencial o paralela y ver incrementado el rendimiento de la operación general. Los pipelines, pueden optimizarse aprovechando los recursos informáticos de los servidores de cálculo empleados; haciendo que sean de gran utilidad en el análisis de datos metagenómicos. El análisis bioinformático de los resultados de la secuenciación de la microbiota, en general, requiere de multitud de pasos que usan muchos recursos de computación. El reciente avance en las técnicas de secuenciación ha desembocado en la producción de grandes cantidades de datos de secuencias en cada experimento que requieren de un procesamiento rápido y abarcable por una infraestructura informática media.

1.3. PIPELINES EN METAGENÓMICA

Uno de los pipelines más completo y avanzado es probablemente SqueezeMeta (Tamames y Puente-Sánchez, 2018). El pipeline ideado y desarrollado por Javier Tamames y Fernando Puente-Sánchez (Departamento de Biología de Sistemas del Centro de Superior de Investigación Científica de Madrid) es un programa que desde una sola línea de comando permite llevar a cabo múltiples tareas encadenando entradas y salidas hasta llegar al resultado final del análisis. Más en detalle este pipeline permite, entre sus funciones principales:

- El procesamiento de múltiples muestras metagenómicas mediante procesos de ensamblado: 1) secuencial, muestra tras muestra; 2) co-ensamblado, juntando todas las muestras en un único proceso; 3) ensamblando de forma secuencial y juntando los

resultados generando una base de datos no redundante. Estos abordajes se elijen en función de la cantidad de datos y el tipo de muestras a analizar.

- Trabajar tanto con lecturas de segunda (Illumina, IonTorrent) y tercera generación (Oxford Nanopore Technologies, Pacific Bioscience).
- La anotación taxonómica de *contigs* (ver Anexo II).
- La anotación funcional de los *contigs*.
- La estimación de la abundancia de genes individuales en cada metagenoma.
- El proceso de *binning* (ver Anexo II) y *bin checking* para la recuperación de genomas individuales.
- La exportación de los datos en formatos estándar para otras herramientas.

Esta plataforma no es la única en el universo de pipelines de análisis bioinformáticos para metagenomas. Los autores comparan sus funciones con otras parecidas (Tabla 1) lo que nos permite tener una idea del nivel de complejidad de los análisis metagenómicos y de los pasos a seguir.
Es importante entender que muchos de los pasos de los pipelines empleados usan programas ya publicados y optimizados. Por ejemplo, en el caso de SqueezeMeta, el pipeline contiene muchos programas escritos por los mismos autores, así como otros ya publicados por terceros con el fin de obtener los mejores resultados y, por supuesto, simplificar la vida de los usuarios que quieren llevar a cabo análisis metagenómicos sin necesariamente ser expertos bioinformáticos.

	MG-Rast (Meyer et al., 2008)	Anvio (Eren et al., 2015)	Smash community (Arumugam et al., 2010)	Humann (Abubucker et al., 2012)	fmap (Kim et al., 2016)	MetaWrap (Uritskiy et al., 2018)	Samsa2 (Westreich et al., 2018)	IMP (Narayanasamy et al., 2016)	Squeeze Meta
Assembly	No	No	Yes	No	No	Yes	No	Yes	Yes
Data source	Reads or contigs	Contigs	Contigs	Reads	Reads or contigs	Contigs	Reads (RNA)	Reads	Reads
Gene prediction	Yes	Yes	Yes	No	No	No	No	Yes	Yes
Function assignment	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
RNA assignment	Yes	Yes	No	No	No	No	Yes	Yes	Yes
Taxonomic assignment	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Gene abundances	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes
Metagomic comparison	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Co-assembly	No	No	No	No	No	Yes	No	Yes	Yes
Binning	No	Support	No	No	No	Yes	No	Yes	Yes
Bin validation	No	Yes	No	No	No	No	No	No	Yes
Local Installation	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Tabla 1. Características de diferentes pipelines de análisis metagenómico, en comparación con SqueezeMeta (Tamames y Puente-Sánchez, 2018).

SqueezeMeta permite su uso en ambientes Linux o Mac y su instalación es simplificada por el uso de plataforma como GitHub o Conda, que permiten clonar todo el pipeline con sus programas accesorios, así como las bases de datos y ficheros de configuración asociados. Luego el usuario ya puede, previa una pequeña tarea de actualización siguiendo las instrucciones, tener ya disponible y operativo el pipeline entero para su uso inmediato.

Otro pipeline muy utilizado y que no necesita instalación es "Metagenomic Rapid Annotations using Subsystems Technology" (MG-RAST). El incentivo principal para la creación del programa fue la obtención de un medio de uso público para analizar las muestras metagenómicas, almacenarlas y eliminar uno de los mayores hándicaps en el análisis: el acceso a herramientas de computación de alto rendimiento (Meyer *et al*., 2008). MG-RAST se basa en un servidor web de código abierto al que se envían los ficheros de secuencias; él mismo se encarga de todo el análisis on-line. El programa, pasa por los procesos de control de calidad y anotación de todas las secuencias devolviendo los resultados de anotación funcional y taxonómica, de forma automática, en un sitio restringido para el propietario del análisis al cual puede acceder por identificación por usuario y contraseña para descargar los resultados en formato de tablas o ficheros de secuencias.
Una de las partes más interesantes del pipeline es la posibilidad de llevar a cabo un análisis comparado entre las muestras disponibles *on-line* de otros estudios y que están en abierto para la comunidad científica. Adicionalmente, el servicio no está limitado a un tipo específico de secuencias ni a una tecnología de secuenciación concreta y el pipeline se presta a la implementación de nuevas etapas de análisis gracias a su estructura modular.
La última versión de MG-RAST, MG-RAST 4.0 (Meyer *et al*., 2019) implementa la posibilidad de compartir los datos obtenidos a través de una interfaz web (Wilke *et al*., 2016) y una API REST (Wilke *et al*., 2015). Gracias a esta innovación es posible reducir el efecto denominado "incertidumbre del método" que conlleva a la obtención de resultados erróneos por la introducción de datos mal procesados por el uso de distintas estrategias de análisis. La posibilidad de reutilizar datos pre-calculados evita este problema.

El pipeline original se implementó en Perl y se sirve diferentes componentes de código abierto: NCBI BLAST (Altschul *et al*., 1990), SQLite (https://www.sqlite.org/index.html), Sun Grid Engine (http://gridscheduler.sourceforge.net), SEED (Overbeek *et al*., 2014).

2. PROCESOS BÁSICOS EN UN ANÁLISIS METAGENÓMICO

2.1. CONTROL DE CALIDAD

Tras el proceso de secuenciación, la curación de las secuencias obtenidas y el control de calidad es un punto fundamental para que un análisis de datos, en general, se considere fiable.
En el control de calidad se lleva a cabo un primer filtrado de las secuencias por calidad se eliminan eventuales adaptadores (ver Anexo II) o secuencias técnicas de origen no biológico y se recortan, eventualmente los extremos donde se detecta una caída de calidad. Este proceso se puede llevar a cabo mediante diferentes programas que han ido evolucionando con el tiempo ofreciendo algoritmos siempre más rápidos y completos.
Uno de los programas empleado para el control de calidad es el fastp (Chen *et al*., 2018). Una de las ventajas de fastp es que permite dividir los ficheros de secuencias en paquetes de un tamaño N (N = 1000, por defecto), cada uno de esos paquetes es procesado paralelamente en un hilo (*thread*). Los hilos se comportan como entornos independientes en los que se procesa cada bloque de datos del paquete (secuencias) para obtener: perfiles de calidad, el contenido de las bases por ciclo de secuenciación, resultados del recorte de adaptadores y recuentos de k-mer (ver Anexo II).
El programa procede al recorte de adaptadores, la detección de los adaptadores se puede producir de forma automática por el programa; o de forma guiada si se le proporcionan secuencias específicas, en solitario o en formato multifasta (ver Anexo I), incluyendo todos los posibles adaptadores a eliminar. En el primer caso, el algoritmo que emplea fastp para la detección de los adaptadores sigue dos premisas: existencia de un único adaptador y que la localización de este se encuentra en el extremo 3' de las lecturas. Ambas premisas se cumplen cuando las lecturas provienen de secuenciadores Illumina. Pese a ello, en el caso de que se obtengan a través de una secuenciación tipo *mate pair* (ver Anexo II) no habrá un único

adaptador, en una sola localización (Data Processing of Nextera Mate Pair Reads on Illumina Sequencing Platforms, 2012). Como consecuencia es necesario especificar qué adaptadores presentarán las secuencias de entrada, recordando que dependen del tipo de librería que se haya usado a la hora de secuenciar. fastp ofrece diferentes comandos para que el usuario introduzca las secuencias de adaptadores de acuerdo con el protocolo de secuenciación empleado, tanto para datos *single-end* (SE) (ver Anexo II), como para *paired-end* (PE) (ver Anexo II), o librerías de adaptadores en formato FASTA (ver Anexo I).
El algoritmo que usa fastp para la detección de adaptadores en datos PE, proviene del empleado por AfterQC, y se basa en la búsqueda de solapamiento entre cada par de lecturas, y el posterior recorte de aquellas bases que queden fuera del solapamiento (aunque sea una única base).

fastp permite llevar a cabo un control de calidad mediante un método de "ventana deslizante" desde el inglés sliding window que se encarga de eliminar aquellas bases que no tengan una calidad suficiente de la cola (3') y de la cabeza de las lecturas. La tecnología de Illumina produce generalmente secuencias de muy alta calidad, pero se encuentra siempre una bajada de dicha calidad en su extremo 3'. Por esta razón es necesario llevar a cabo dicho control. El funcionamiento del proceso consiste establecer ventanas de un tamaño dado (WS); las ventanas se moverán desde uno de los extremos de las lecturas, dependiendo de si se usa "*-cut_front*" (desde 5') o "*-cut_tail*" (desde 3') (Chen *et al*., 2018).

Existen otras funciones como "global trimming" o "recorte global", que efectúa un recorte en las lecturas de la parte anterior a la cola, que son de peor calidad al ir sucediéndose los ciclos de secuenciación; recorte de cola poliG, esta eliminación es necesaria en lecturas de NextSeq y NovaSeq que emplean sistemas de dos colores para representar las cuatro bases, con el paso de los ciclos de secuenciación las señales colorímetricas se pueden malinterpretar dando como resultado la interpretación de timina (T) y citosina (C) como guanina (G); recorte de cola poliX, donde X es cualquier base; y pre-procesamiento UMI. La tecnología UMI es útil para eliminar el ruido de fondo como las duplicaciones, el sistema se basa en la generación de lecturas de consenso con una calidad bastante elevada. El pre-procesamiento UMI ya se había integrado para archivos FASTQ (ver Anexo I) con herramientas como umis (Kirchner, 2018) y UMI-tools (Smith *et al*., 2017). No obstante, fastp es tres veces más rápido que estos programas.

2.2. ENSAMBLADO

Posterior al control de calidad, el siguiente paso consiste en el ensamblado de las lecturas para intentar extender su longitud y reconstruir *contigs* más largos para su posterior anotación. Un ensamblaje genera una estructura jerárquica de datos que "ordena" los datos "problema" (input) de forma que el producto final es la reconstrucción de fragmentos más largos de secuencias. Las lecturas son ordenadas en *contigs*, y estos, en *scaffolds* (ver Anexo II) o *supercontigs*. Mientras que los *contigs* se asemejan a las piezas de un puzzle: representan el resultado del alineamiento múltiple de lecturas que proporciona secuencias consenso de partes del genoma a ensamblar; los scaffolds son las pistas que ayudan al montaje del puzzle, definiendo el orden y la orientación de los primeros. Las características a tener en cuenta en el proceso son el tamaño de los *contigs* y *scaffolds*, y la calidad del ensamblaje. Este último punto es algo abstracto y difícil de medir, aunque ya existen herramientas de validación como la validación *mate-pair* o la localización de lecturas repetidas (Phillippy *et al*., 2008 y Miller *et al*., 2010).

El ensamblaje de secuencias se puede dar siguiendo dos métodos distintos. El primer enfoque se refiere al ensamblaje con genoma de referencia. Dicha estrategia consiste en emplear genomas de referencia como guía para el proceso. Algunos paquetes de *software* de ensamblaje bastante conocidos son el Modular Open-Source Assembler (AMOS), que tiene un enfoque modular y ofrece herramientas para el ensamblaje *de novo* (Minimus, Minimo y Bambus 2) (Koren *et al*., 2011); o MIRA (Chevreux *et al*., 2004). El segundo, ensamblaje *de novo* necesita mayores recursos de computación y se basa en los diagramas de Brujin, ejemplos de herramientas que realizan este tipo de ensamblaje son SPAdes y MEGAHIT (Li *et al*., 2015). Aunque existen

estos dos enfoques, el primero se encuentra con un gran inconveniente cuando se intenta aplicar al campo de la metagenómica: la gran mayoría de los microorganismos presentan grandes diferencias intra- e inter-especie. Por lo tanto, un ensamblaje con un genoma de referencia dificultaría la formación de *contigs* cuando las lecturas pertenezcan a secuencias "no clonales" (Thomas *et al*., 2012). Como consecuencia, el enfoque habitual en los pipelines metagenómicos para el proceso de ensamblaje es el tratamiento *de novo*, y particularmente, en el script aquí presentado se ha elegido SPAdes.
SPAdes es una herramienta que mejora ensambladores de última generación para secuenciación de célula única (SCS) y para datos provenientes de muestras multicelulares, como E + V-SC (Chitsaz *et al*., 2011), SoapDeNovo (Li *et al*., 2010) o Velvet (Zerbino y Birney, 2008). Además, se ha demostrado que tiene una gran precisión (Al-Okaily, 2016) y calidad (Liu *et al*., 2016).
SPAdes es un ensamblador de código abierto que basa su funcionamiento en la síntesis de grafos de Brujin (ver Anexo II), soluciona el problema de falta de precisión que presentan los Gráfos Emparejados *de* Brujin (PDBG) (ver Anexo II), centrándose en medir con exactitud la variación que existe en las distancias de las bi-lecturas (en concreto entre el par de k-mers de cada lectura o k-bimers), mediante un algoritmo de ajuste de estos k-mers y diferentes gráficos emparejados de ensamblaje inspirados en el método PDB (*Bankevich et al.,* 2012*).*
En el contexto de el análisis de datos metagenómicos, SPAdes posee un ensamblador orientado precisamente a la metagenómica: metaSPAdes, el cual se diseñó con el objetivo de paliar la diversidad presente en los microorganismos. Adicionalmente, implantó características como novedosos métodos para mejorar el ensamblaje de muestras con una mezcla heterogénea de cepas, o algoritmos de corrección de errores y construcción de grafos de ensamblaje. En consecuencia, y tras comparar diferentes ensambladores metagenómicos como IDBA-UD (Peng *et al*., 2012), Ray-Meta (Boisvert *et al*., 2012) y MEGAHIT (Li *et al*., 2015) con metaSPAdes se constata que este último ofrece un ensamblaje más completo en una amplia variedad de muestras, superando así a sus competidores (Nurk *et al*., 2017).

2.3. ANOTACIÓN

El siguiente paso consiste en la anotación de las secuencias, tanto estructural como funcional. Esta etapa es fundamental, lo que se ha recogido a lo largo de los años por diversas publicaciones y se ejemplifica con las siguientes afirmaciones: "The value of the genome is only as good as its annotation" (Stein, 2001), o "Genome annotation is the process of identifying and labeling all the relevant features on a genome sequence" (Richardson y Watson, 2013).
Gracias al proceso de ensamblaje se obtienen *contigs* y *scaffolds*, estos representan las bases genómicas de los microorganismos que se hallan en las muestras estudiadas.

El objetivo de la anotación es relacionar los datos de secuencias con diferentes genes, la función que realizan, las proteínas que codifican y, a su vez, con los microorganismos de lo que potencialmente proceden. El resultado del proceso depende del tipo de anotación que se aborde, existen diferentes clasificaciones aquí se exponen dos de ellas.

Según la información que se quiera obtener existen dos abordajes, la anotación estructural y la anotación funcional.

- Anotación estructural.

Como su nombre indica este tipo de anotación pretende "dibujar" – predecir y localizar – los genes, y su estructura, (exones, intrones, ORFs [ver Anexo II], transposones, …) que se encuentran en las secuencias problema. Estas secuencias suelen corresponder con los *contigs* identificados en el proceso de ensamblaje (uniones de lecturas). Además de genes, también es posible la identificación de proteínas que son codificadas por estos. Es decir, la anotación estructural se puede dar a diferentes niveles, a nivel de nucleótidos, a nivel de proteínas, a nivel de dominios estructurales o de funciones.

Existen diferentes procedimientos para realizar una anotación estructural: método *ab initio*, identificación a partir de la secuencia de DNA; método basado en homología, identificación a través del uso de conceptos de la conservación evolutiva; y método híbrido (Mathé *et al*., 2002.) que corresponde a una mezcla de ambos.

- Anotación funcional.

En cambio, la anotación funcional pretende inferir la función bioquímica, biológica, expresión, regulación e interacciones pertenecientes a las estructuras genómicas que se identifican gracias a la anotación estructural. Las características estudiadas gracias a la anotación funcional se relacionan íntimamente con los términos GO (Gene Ontology), cuya página web oficial es http://www. geneontology.org/ (Primmer *et al*., 2013) que clasifican al gen de acuerdo a las siguientes ccategorías:función molecular, proceso biológico en el que interviene y componente celular o espacio celular que alberga al producto génico. A consecuencia, la segunda anotación (funcional) suele acompañar a la primera. Una de las herramientas más conocidas que se emplean en este tipo de anotación (basada en homología) es la familia de programas BLAST (Altschul *et al*., 1990 y Altschul *et al*., 1997).

Según la metodología empleada se encuentran otros dos enfoques diferentes, la anotación manual y la anotación automática.

- Anotación manual.

La anotación manual se realiza a través de expertos con un conocimiento profundo de las secuencias ensambladas, en particular, y de la estructura de secuencias, en general.
Al ser una interpretación humana los resultados suelen ser más precisos y de mayor calidad. No obstante, se convierte en una anotación muy lenta, y puede ser que los diferentes anotadores envueltos en el proceso lleguen a conclusiones diferentes e incluso contradictorias (Potter *et al*., 2004).

- Anotación automática.

En cambio, la anotación automática se produce mediante un conjunto de herramientas informáticas encargadas de llevar a cabo los diferentes pasos que requiere el proceso; *grosso modo* se podrían dividir en dos procedimientos: predicción génica (anotación estructural) y asignación de funciones biológicas y relaciones taxonómicas (Thomas *et al*., 2012). Existen una gran variedad de programas que se pueden emplear para las diversas tareas: MetaGeneMark (Zhu *et al*., 2010), FragGeneScan (Rho *et al*., 2010), MetaGeneAnnotator (MGA)/Metagene (Noguchi *et al*., 2008), Orphelia (Hoff *et al*., 2009). Algunas de las bases de datos más empleadas para la anotación funcional son KEGG (Kanehisa *et al*., 2004), eggNOG (Jensen *et al*., 2008), COG/KOG (Tatusov *et al*., 2003), PFAM (Finn *et al*., 2010), and TIGRFAM (Selengut *et al*., 2007).
Actualmente ya existe una gran variedad de anotadores automáticos que abordan el proceso desde diferentes estrategias: búsqueda de ORFs, tablas de uso de codones, contenido en guanina-citosina (GC), alineamiento de cDNA (ver Anexo II) y marcadores de secuencia expresada (EST) (ver Anexo II), comparación con bases de datos, etc. Este hecho se traduce en que las mismas secuencias pueden obtener anotaciones distintas según el programa que se emplee; ya que, cada uno emplea unas evidencias para inferir las estructuras génicas y sus funciones.

Cada anotador tiene sus propias particularidades en cuanto a estrategia, finalidad, tiempos de ejecución y tipos de genoma en los que se especializa: servidores como RAST que es un anotador de genomas pertenecientes a arqueas y bacterias, capaz incluso de construir modelos de redes metabólicas, en un tiempo medio de 12 a 24 horas (Aziz *et al*., 2008). Blast2GO, anotador funcional de nuevas secuencias y analizador de datos pre-anotados, del que hay disponible documentación online (https://www.blast2go.com/blast2go-pro). xBASE2, base de datos para el estudio de secuencias de genomas bacterianos (genómica comparada), su tiempo de procesamiento es de unas pocas horas (Chaudhuri *et al*., 2008).

Prodigal (PROkaryotic DYnamic programming Gene-finding ALgorithm), algoritmo de predicción génica enfocado en genomas bacterianos que ofrece una gran sensibilidad y precisión, al disminuir los falsos positivos (Hyatt *et al*., 2010). GeneMark que es una familia de anotadores entre los que se destacan GeneMarkS (Besemer *et al*., 2001), GeneMark.hmm (Lukashin y Borodovsky, 1998) y GeneMark-ES (Lomsadze *et al*., 2005), cada uno especializado en un campo de la anotación, predicción en bacterias y arqueas, predicción gracias a modelos heurísticos y predicción de genomas nuevos de eucariotas, respectivamente. E incluso el NCBI tiene un anotador automático propio. No obstante, aún se les considera anotadores demasiado lentos y poco adecuados para aplicaciones que requieren de un elevado rendimiento y privacidad.

2.3.1. Anotación estructural

Un programa de especial interés es Prokka (Seemann, 2014) para el proceso de anotación. Prokka es un paquete que implementa diferentes herramientas para producir anotaciones fiables en muy poco tiempo, unos 10 minutos en el caso de genomas bacterianos típicos empleando un ordenador de sobremesa. Es posible realizar anotaciones funcionales con Prokka; pero en este caso, el objetivo es obtener archivos con datos de anotación estructural para su posterior procesado.

El input que recibe este programa consiste en archivos formato FASTA de *contigs.*
Las diferentes herramientas de software que emplea Prokka y la función que cumple cada una de ellas se pueden observar en la Tabla 2.

Tool (reference)	Features predicted
Prodigal (Hyatt 2010)	Coding sequence (CDS)
RNAmmer (Lagesen *et al*., 2007)	Ribosomal RNA genes (rRNA)
Aragorn (Laslett and Canback, 2004)	Transfer RNA genes
SignalP (Petersen *et al*., 2011)	Signal leader peptides
Infernal (Kolbe and Eddy, 2011)	Non-coding RNA

Tabla 2. Herramientas de predicción de características utilizadas por Prokka (Seemann, 2014).

Este programa produce diez archivos output diferentes que llevarán el mismo prefijo definido en la línea de comando del *script.* Cada uno de los archivos contiene diferentes resultados que se especifican en la Tabla 3.

Suffix	Description of file contents
.fna	FASTA file of original input contigs (nucleotide)
.faa	FASTA file of translated coding genes (protein)
.ffn	FASTA file of all genomic features (nucleotide)
.fsa	Contig sequences for submission (nucleotide)
.tbl	Feature table for submission
.sqn	Sequin editable file for submission
.gbk	Genbank file containing sequences and annotations
.gff	GFF v3 file containing sequences and annotations
.log	Log file of Prokka processing output
.txt	Annotation summary statistics

Tabla 3. Descripción de los archivos de salida de Prokka (Seemann, 2014).

2.3.2. Anotación funcional

Para continuar con la anotación funcional se emplean dos procesos distintos: alineación de los aminoacidos (archivos en formato FASTA, ".faa") provenientes del proceso PROKKA contra las bases de datos KEGG y eggNOG. Dicha anotación se realiza mediante la herramienta DIAMOND (Buchfink *et al.*, 2015). Contra la base de datos Pfam se elige el programa HMMER para comparar, en este caso, solamente secuencias de aminoácidos introduciendo un nuevo concepto: los modelos ocultos de Markov (HMMs), estos describen una distribución de probabilidad sobre un número de secuencias potencialmente infinito" (Eddy, 1998).

DIAMOND es un algoritmo de alineamiento de datos (proteínas y DNA) provenientes de NGS que emplea un enfoque de doble indexado. Más concretamente, plantea dos listas de *seeds* (tramos cortos de secuencia) una en la secuencia a mapear y otra en la secuencia de referencia. Este alineamiento se produce contra diferentes bases de datos como NCBI-nr (Benson *et al.*, 2005) o las ya nombradas KEGG y eggNOG.

Además, para aumentar la velocidad del proceso con respecto a BLASTX (Altschul *et al.*, 1990), que se ha considerado la "herramienta de oro" para estos procesos, sin perder la sensibilidad, DIAMOND divide las secuencias en *seeds* más largas (*spaced seeds*), de las cuáles solo utiliza para el mapeo un *subset* de posiciones que se caracterizan por su forma (*shape*) y peso (*weight*). Se emplea el algoritmo Smith-Waterman (Smith y Waterman, 1981) para puntuar los alineamientos y escoger *seeds* óptimas.

Por otro lado, "HMMER es un paquete de software que proporciona perfiles de modelos probabilistas de familias de dominios de proteínas y DNA que se denominan perfiles de modelos de ocultos de Markov (HMMs) o simplemente perfiles" (Eddy, 2020).
Un HMMs es un modelo probabilístico (Figura 5), para procesos de Márkov o procesos estocásticos (magnitudes aleatorias que varían con el tiempo) que se rigen por la propiedad de Markov, que consisten en que el estado del modelo en el instante actual (t+1) depende del estado del modelo en el instante anterior (t) y solo se observa una secuencia x - una secuencia de símbolos- , mientras que la secuencia de estados visitados – que generan esos símbolos - permanece oculta (Universitat Politècnica de València, 2017, 35s).

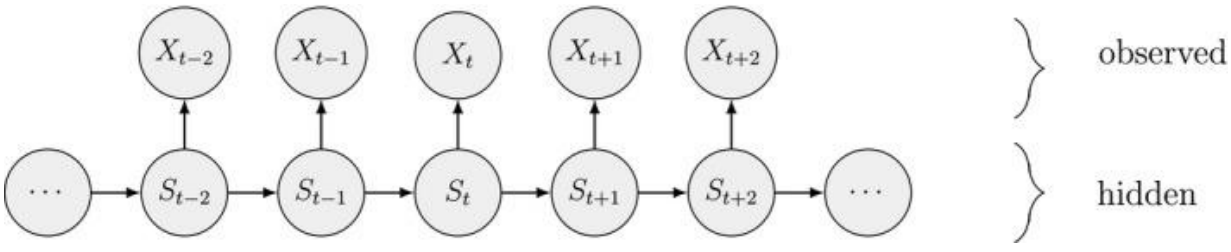

Figura 5. Estructura básica de dependencia de un modelo oculto de Márkov (HMM) con una secuencia observada que surge de una secuencia no observada (McClintock *et al.*, 2020).

Matemáticamente un HMM es una lista ordenada de elementos (tupla), cada uno de ellos define una parte de este modelo. Sus componentes incluyen conjunto finito de estados (Q), un conjunto de valores observables o "alfabeto" (*V*), y diferentes probabilidades que definen la probabilidad de pasar de un estado a otro dentro del sistema, para dar una pequeña pincelada de sus componentes (Mohamed y Gader, 2000 y Dymarski, 2011).

La ventaja que confiere el uso de HMM es la información adicional que se puede obtener de la comparación entre secuencias y un modelo (perfil) estadístico que describe una familia de proteínas, mediante la detección de sitios de secuencia eliminados (que afectan o no la secuencia), inserciones, conservación de residuos específicos de familia de proteínas o variación de aquellos que no son esenciales.

Los perfiles obtenidos sirven para la anotación de secuencias, búsqueda de homólogos en bases de datos y alineaciones múltiples. HMMER emplea algoritmos de ensamblado (*ensemble algorithms*) que consideran todos los alineamientos posibles y proporciona un grado de probabilidad para cada residuo alineado. Como aspecto positivo, este tipo de algoritmos es capaz de calcular las para la secuencia y no para el alineamiento. Esto es, la razón de momios (*odds ratio*) para cada secuencia no es la puntuación óptima de alineación, sino que las puntuaciones se suman sobre el ensamblado posterior. Por lo tanto, HMMER se centra en el cálculo completo de la "posibilidad de que se produzca el alineamiento, no en el aproximado"
El paquete de software de HMMER contiene varios programas, entre ellos *hmmsearch* que "busca coincidencias para un único perfil HMM dentro de una base de datos de secuencias" (Eddy, 2020).

2.3.3. Bases de datos

En cuanto a las bases de datos elegidas, KEGG (Kyoto Encyclopedia of Genes and Genomes) es un recurso bioinformático al que se puede acceder mediante el enlace a KEEG Home (https://www.genome.jp/kegg/), en concreto una base de datos, para comprender las funciones y utilidades de sistemas biológicos a diferentes niveles (célula – organismo – ecosistema) gracias a información procedente de la secuenciación de su genoma (Kanehisa *et al*., 2004); y "eggNOG (evolution gene genealogy database: Non-supervised orthologous groups) es una base de datos de dominio público que alberga información sobre las relaciones ortólogas entre genes, su historia evolutiva y anotación funcional" (Huerta-Cepas *et al*., 2019). La base de datos está enfocada en proporcionar anotaciones funcionales exhaustivas para los genes ortólogos inferidos; asignaciones ortológicas basadas en una resolución jerarquizada y análisis filogenéticos; y consulta de cientos de genomas distintos (cubren los 3 dominios de la vida y virus) para hacer las predicciones.
Pfam está enfocada a la clasificación de secuencias de proteínas en familias y dominios - cada una de ellas definida por dos alineamientos - "es una base de datos basada también en perfiles de HMMs que combina alta calidad y minuciosidad de análisis" (Sonnhammer *et al*., 1997). Cada familia proteica tiene su propio perfil de HMM, a partir del que se procederá a hacer un alineamiento múltiple con las secuencias de entrada.
Dentro de la propia base de datos existen dos clasificaciones: Pfam-A y Pfam-B. En la primera de ellas las anotaciones se consideran de alta calidad y se encuentran curadas a mano. En ocasiones, es posible que Pfam-A construya varias entradas para representar diferentes familias - pertenecientes a superfamilias proteicas - que se organizan en clanes (Finn *et al*., 2006) porque un único perfil HMM no es capaz de establecer su homología. Ante esta posibilidad aparece Pfam-B que se trata de una base de datos cuyas secuencias derivadas de la base de datos PRODOM (Servant *et al*., 2002), se encuentran anotadas automáticamente y cubre aquellas secuencias que no aparecen en Pfam-A (Finn *et al*., 2014). Tanto Pfam-A como Pfam-B tienen una base de datos de fragmentos y una de dominios completos, la última se basa en modelos globales de HMMs y es más completa.

La cobertura de secuencias que ofrece actualmente Pfam (33.0) se encuentra sobre el 77%, es decir un 77% de las muestras analizadas tendrán al menos una coincidencia en la base de datos (Mistry *et al*., 2021).

2.4. MAPEO

Otro tipo de análisis corresponde con el mapeado de las secuencias. Los archivos necesarios para mapear son un genoma de referencia y los archivos de secuencias a mapear.

En general existen dos estrategias de algoritmos de alineamiento de lecturas provenientes de NGS:

- **Aproximación por semillas** (seed approach) o *hashing*: consiste en la división de las secuencias a mapear en partes de pequeño tamaño – k-mer con un tamaño de 11 nucleótidos por defecto en BLAST – para buscar *matches* con una identidad del 100% entre estas y la secuencia del genoma de referencia. Las coincidencias se denominan semillas. Posteriormente, se emplea el algoritmo Smith-Waterman (SW) para alinear aquellas lecturas que no hayan conseguido esa identidad del 100%. La aproximación por semillas consigue unos alineamientos locales prácticamente óptimos (Li y Homer., 2010) y como ejemplo de algunas herramientas que emplean la estrategia denominada seed and extend se destacan BLAST, Eland (http://www.gersteinlab.org/proj/chip-seq-simu/eland.html), SOAP (Li *et al*., 2008a), MAQ (Li *et al*., 2008b), CloudBurst (Schatz, 2009), SeqMap (Jiang y Wong, 2008), SHRiMP (Rumble *et al*., 2009) o Novoaling (http://www.novocraft.com/products/novoalign/). Si bien es cierto que todas tienen la base común, difieren en aspectos como la sensibilidad, la rapidez de procesamiento o el uso de diferentes métodos de *seed and extend* que se pueden dividir en tres sistemas distintos: *spaced seed* que mejora la sensibilidad reduciendo el número de matches consecutivos obligatorios en el mapeo, filtro q-gram que favorece el mapeo cuando existen *gaps*, o vectorización para hacer más dinámico el funcionamiento de SW.

- **Transformación de Burrows-Wheeler (BWT):** BWT es un algoritmo cuya principal aportación es la compresión de datos. Por lo que reduce considerablemente la memoria necesaria para el proceso de alineamiento, siendo el requerimiento muy alto en la anterior aproximación. La aproximación BWT es muy útil cuando se quiere mapear secuencias con una identidad muy elevada (99% - 100%). Las herramientas que usan la BWT son muy rápidas y tan sensibles como sus homólogas basadas en el *seed aproach*. No obstante, su rendimiento disminuye exponencialmente con el número de *mismatches*. Las herramientas basadas en BWT entran en el grupo de aquellas que usan algoritmos basados en *suffix/prefix tries*. El mapeo mediante la aproximación de BWT se centra en relacionar al índice FM (ver Anexo II) con el denominado *trie* (árbol de sufijos/partes de secuencia de una longitud determinada) siendo esta aproximación la predominante. Bowtie y BWA son paquetes de software enmarcados en este tipo.

Bowtie es una herramienta de mapeo indicada para archivos FASTQ y FASTA que funciona creando un índice permanente de referencia que se emplea a lo largo de todo el proceso de alineamiento y sirve como guía, tiene una sensibilidad igual o algo inferior a otros paquetes de software como MAQ o SOAP pero posee un mayor rendimiento – al mapear el genoma humano y otros organismos modelo como el perro, chimpancé o pollo – además es capaz de funcionar en un ordenador de menos de 2GB de RAM y permite el uso simultáneo de varios núcleos de procesamiento para aumentar su velocidad de trabajo (Langmead *et al*., 2009).

BWA (Burrows-Wheeler Alignment tool) es un paquete de software para el mapeo de secuencias (con un bajo nivel de divergencia) contra una base de datos de referencia amplia. El paquete consta de 3 algoritmos distintos: BWA-backtrack está diseñado especialmente para el mapeo de lecturas provenientes de Illumina con una longitud máxima de 100bp. Por otro lado, BWA-MEM y BWA-SW se emplean en el mapeo de lecturas más largas, la principal diferencia entre ambos es que el primero de ellos es capaz de obtener resultados de mayor calidad a una velocidad mayor. E incluso, BWA-MEM tiene mejores resultados en el mapeo de lecturas de Illumina que BWA-backtrack (Li, 2012). Los tres algoritmos necesitan de la formación inicial de un índice FM para el genoma de referencia y posteriormente cada uno implementa otros tipos de aproximaciones según su finalidad.

BWA-MEM trabaja con algoritmos que emplean alineamientos de semilla con la máxima coincidencia exacta (MEMs) y luego extiende las semillas mediante el algoritmo Smith-

Waterman. Mediante este comando ("bwa mem") se obtiene un alineamiento local. En el caso de las secuencias largas, se realizan múltiples alineamientos primarios para diferentes partes de una secuencia problema.
BWA es una herramienta que, con secuencias cortas, es capaz de dar una respuesta más rápida que otros programas como MAQ. Basándose en los criterios de porcentaje de mapeo/tasa de error de alineamiento, lecturas alineadas/número de lecturas mapeadas por pares, y fracción de lecturas mal mapeadas si se proporciona una secuencia de referencia perteneciente a una especie divergente, BWA supera a programas como Bowtie o SOAP (Li y Durbin, 2009).
Existe, además, un editor de archivos en formato SAM, BAM o CRAM denominado Samtools (Li *et al*., 2009) que se presenta de gran utilidad para su uso conjunto con BWA que produce archivos con el mencionado formato.

2.5. EXPRESIÓN DIFERENCIAL

El último paso del procesamiento consiste en el análisis de diferencial de coberturas entre muestras.
Este tipo de abordaje se usa generalmente en los análisis de expresión diferencial entre muestras. Los algoritmos que se emplean permiten cuantificar la cantidad de secuencias que recubren cada anotación en cada muestra permitiendo así un análisis diferencial de muestras genómicas o metagenómicas entre ambientes o condiciones.
El concepto de expresión diferencial se ha estudiado y definido en numerosas publicaciones, por ejemplo:
"La expresión génica diferencial se define como la decodificación celular de la información contenida dentro del material genético (ácidos nucleicos) para la elaboración del producto génico necesario para el buen funcionamiento del organismo" (Hillis *et al*., 2020).
En otras palabras, el análisis diferencial de la expresión hace referencia al estudio de la expresión de distintos genes en distintos momentos (bajo diferentes condiciones) en un mismo organismo. El hecho de conocer cuándo, cómo y bajo qué circunstancias se expresa un gen es una tarea ardua y de gran importancia para el estudio de la función biológica de estos (DeRisi *et al*., 1997).
En un primer momento, los estudios de la expresión diferencial se realizaban mediante técnicas como los microarrays. Hasta el momento en el que las NGS permitieron que la secuenciación fuese económicamente accesible, este hecho condujo al desarrollo del RNA-seq (ver Anexo II) como método usual en este tipo de experimentos. El RNA-seq es más sensible que, por ejemplo, los microarrays (Zhao *et al*., 2014) siendo una técnica que hace uso de la secuenciación masiva para observar la cantidad y la presencia del RNA en un instante determinado en una muestra biológica (Wang *et al*., 2009).

En concreto para este cometido se destaca *htseq-count*.
HTSeq es una biblioteca de Python, de código abierto, que se centra en el análisis de datos provenientes de secuenciación de alto rendimiento (HTS). El paquete de HTSeq ofrece dos herramientas diferentes: *htseq-qa*, que analiza la calidad de las lecturas de secuenciación, y *htseq-count*, que analiza los datos de RNA-seq para el análisis de la expresión diferencial.
El funcionamiento de *htseq-count* consiste en contar, para cada uno de los modelos de genes de un archivo de anotación GTF o GFF (ver Anexo I), las lecturas alineadas que se superponen con sus exones de un archivo de mapeo, como por ejemplo ficheros SAM/BAM (ver Anexo I). Solo se tienen en cuenta aquellas que se asignan a un gen único e inequívoco (Anders *et al*., 2015). Posteriormente este recuento se procesa mediante programas como edgeR (Robinson *et al*., 2010) para el análisis génico de la expresión diferencial. Esta herramienta fue una de las primeras en este campo. Con el tiempo se han ido desarrollando otras como featureCount (Liao *et al*., 2014) o Cufflinks1 y 2 (Trapnell *et al*., 2010 y Trapnell C. *et al*., 2012). En un estudio de comparación empírica entre varias de estas herramientas se consideró que htseq-count es menos dependiente de la elección de programas de mapeo que sus competidoras mostraban diferentes resultados según si la elección variaba (Fonseca *et al*., 2014).

3. RESULTADOS

3.1. NEXTFLOW (DSL1)

Los avances experimentados por las tecnologías de secuenciación han desembocado en un mayor número de experimentos ómicos que requieren de análisis reproducibles de cantidades ingentes de datos. Las primeras aproximaciones de herramientas bioinformáticas -*pipelines*-, que se desarrollaron para este cometido, sufren de inestabilidad numérica (*numerical instability*) cuando el mismo *pipeline* se usa en diferentes plataformas, de forma que se imposibilita la reproducibilidad de los trabajos (Garijo *et al*., 2013).

Nextflow nace como lenguaje para la construcción de pipelines mejorando el uso de los recursos informáticos, controlando el flujo de datos de entrada y salida y manteniendo un registro de todas las actividades desarrolladas al fin de poder seguir el flujo de trabajo mismo, corregir errores eventuales y mantener un registro de los comandos ejecutados. Nextflow es un sistema de gestión de flujos de trabajo que utiliza distintas infraestructuras informáticas como Docker, Conda, Singularity, etc. para el manejo a gran escala mediante el uso de los que se conocen como "contenedores". Estos últimos son entornos de computación configurados de tal forma que solamente admiten las entradas y salidas de datos para lo que se han desarrollado. (Di Tommaso *et al*., 2017).

Las características que hacen de Nextflow una herramienta útil y flexible para la creación de pipelines son la posibilidad integración en repositorios de software como BitBucket, cuya página web se corresponde con la dirección https://bitbucket.org/ o GitHub, https://github.com/, un soporte nativo para sistemas en la nube. Además, se basa en el uso de contenedores multiescala, y de la paralelización de los procesos que se conectan mediante canales. Todas estas características mejoran la reproducibilidad computacional y posicionan a Nextflow como un lenguaje de dominio específico que adelanta a sus competidores como se puede apreciar en la Tabla 4.

Workflow	**Nextflow**	**Galaxy**	**Toil**	**Snakemake**	**Bpipe**
Platform[a]	**Groovy/JVM**	**Python**	**Python**	**Python**	**Groovy/JVM**
Native task support[b]	Yes (any)	No	No	Yes (BASH only)	Yes (BASH only)
Common workflow language[c]	No	Yes	Yes	No	No
Streaming processing[d]	Yes	No	No	No	No
Dynamic branch evaluation	Yes	?	Yes	Yes	Undocumented
Code sharing integration[e]	Yes	No	No	No	No
Workflow modules[f]	No	Yes	Yes	Yes	Yes
Workflow versioning[g]	Yes	Yes	No	No	No
Automatic error failover[h]	Yes	No	Yes	No	No
Graphical user interface[i]	No	Yes	No	No	No
DAG rendering[j]	Yes	Yes	Yes	Yes	Yes
Container management					
Docker support[k]	Yes	Yes	Yes	No	No
Singularity support[l]	Yes	No	No	No	No
Multi-scale containers[m]	Yes	Yes	Yes	No	No
Built-in batch schedulers[n]					
Univa Grid Engine	Yes	Yes	Yes	Partial	Yes
PBS/Torque	Yes	Yes	No	Partial	Yes
LSF	Yes	Yes	No	Partial	Yes
SLURM	Yes	Yes	Yes	Partial	No
HTCondor	Yes	Yes	No	Partial	No
Built-in distributed cluster[o]					
Apache Ignite	Yes	No	No	No	No
Apache Spark	No	No	Yes	No	No
Kubernetes	Yes	No	No	No	No
Apache Mesos	No	No	Yes	No	No
Built-in cloud[p]					
AWS (Amazon Web Services)	Yes	Yes	Yes	No	No

Tabla 4. Comparación entre Nextflow y distintas herramientas de gestión de flujos. Entre las diversas herramientas de gestión de flujos de trabajo similares a Nextflow, Bpipe11 es probablemente la más "parecida" a Nextflow. Sin embargo, uno de los problemas de Bpipe es la falta de soporte para el uso de contenedores multiescala, que sí está presente en Nextflow y otros gestores de flujos de trabajo como Galaxy y Toil.

[a]La tecnología y el lenguaje de programación en el que se implementa cada *framework*. [b]Capacidad del *framework* para soportar la ejecución de comandos y *scripts* nativos sin reimplementar los procesos originales. [c]Soporte para la especificación CWL. [d]Capacidad de procesar las entradas/salidas de las tareas como un flujo de datos. [e]Soporte para plataformas de gestión y compartición de código, como GitHub, BitBucket y GitLab. [f]Soporte para módulos, subflujos o composiciones de flujos de trabajo. [g]Capacidad de rastrear cambios en el pipeline y ejecutar diferentes versiones en cualquier momento. [h]Soporte para el manejo automático de errores y mecanismo de reanudación de la ejecución. [i]Implementación de una interfaz de usuario gráfica (GUI) para interactuar con el *pipeline*. [j]Capacidad de visualizar el gráfico de las dependencias y ejecuciones de las tareas. [k]Soporte integrado para la tecnología de contenedores Docker. [l]Soporte integrado para la tecnología de contenedores Singularity. [m]Capacidad de gestionar la ejecución de múltiples instancias de contenedores en un clúster HPC o en la nube. [n]Capacidad de generar las ejecuciones de las tareas del pipeline a través de un programador de lotes del cluster sin necesidad de scripts o comandos personalizados. Hay que tener en cuenta que, aunque este soporte no está incorporado en Snakemake, simplemente requiere que el usuario proporcione los comandos de control de trabajo específicos del clúster. [o]Capacidad de generar las ejecuciones de las tareas de *pipeline* a través de un clúster distribuido. [p]Capacidad de desplegar y ejecutar flujos de trabajo distribuidos aprovechando las características distintivas de la infraestructura de la nube, como las asignaciones de recursos elásticos y las instancias puntuales. Las negritas indican una lista de componentes o aplicaciones similares y compatibles (Di Tomaso *et al*., 2017).

Un *pipeline* implementado en Nexflow puede contener cualquier lenguaje - incluso varios diferentes – compatible con Linux, como Python o Perl. Los procesos dentro de dicho *pipeline* estarían conectados mediante *canales* o colas FIFO asíncronas, ejecutándose en paralelo e independientemente cuando el input que requieren esté disponible. Un mismo proceso puede tener varios *inputs* y *outputs,* y diferentes procesos pueden tener un mismo *input* que llega a través de canales diferentes. En Nextflow se encuentran dos tipos distintos de canales:

- Canal de colas (*queue channel*): cola FIFO no bloqueante y unidireccional. Este tipo de canal conecta dos procesos, es decir, el canal solo se puede usar una única vez como salida y solo una vez como entrada. En el caso de necesitar que se conecten varios procesos es posible duplicar – triplicar, cuadruplicar, … – un canal de colas mediante el operador *into.*

- Canal de valores (*value channel*) o canal *singleton*: canal que contiene un solo valor, por lo que se puede emplear ilimitadamente como input sin ser consumido.

Por otro lado, Nextflow crea directorios únicos y temporales para cada proceso del *pipeline*, lo que implica que no es necesario organizar los *outputs* y facilita la gestión de estos, ya que no se requiere una estructura de directorios organizada o nombrar los archivos de forma estática para su identificación inequívoca. Más aún, en cada proceso no es imprescindible nombrar cada archivo de salida, pues al existir directorios únicos para cada proceso, no se produce la superposición entre archivos. También, es posible relacionar los outputs con metadatos mediante el operador *tuple* para evitar incluirlos en el nombre del archivo de salida. (NEXTFLOW'S DOCUMENTATION!, 2020).

Se pueden encontrar diversos enfoques de flujo de datos, como el enfoque Make (Mölder *et al*., 2021) o de grafo acíclico dirigido (DAG) (ver Anexo II). Ambos enfoques tienen sus limitaciones, en el caso de Make se requiere de una estimación y resolución de dependencias de software previa, y los DAG necesitan de la disponibilidad de mucho espacio de almacenamiento. Nexflow emplea una estrategia de flujo denominada *top to bottom*, la cuál sigue un flujo natural. El modelo *top to bottom* sortea los requerimientos de los enfoques anteriores, ya que, el grafo que sigue cada proceso de Netflow no necesita de cálculos ni ser almacenado (Di Tommaso *et al*., 2017).

El *pipeline* presentado en esta tesis viene implementado en Nexflow y aborda todos los procesos descritos en la memoria para el análisis de datos metagenómicos. La implementación en dicho lenguaje permite la instalación e integración del *pipeline* en sistemas Linux y Mac mediante una

sola línea de comando siendo integrado en repositorios públicos, en este caso concreto, en GitHub. Además, permite la optimización de recursos de computación en análisis metagenómicos paralelizando y aprovechando el uso de los procesadores y de la memoria RAM.

3.2. CONSIDERACIONES INICIALES

Para ejecutar el pipeline objeto de esta tesis – *main.nf* – , se tendrá que tener instalado previamente el ambiente Nextflow siguiendo las instrucciones reportadas en la página web: https://www.nextflow.io/. El requerimiento básico es que se tenga previamente instalado una versión de Java superior a la número 8.
main.nf se puede clonar en el ordenador del usuario mediante el enlace: https://github.com/gdauria/nf-mtg-engine.git. Dentro del repositorio se encuentran todos los archivos y carpetas necesarias para que el *pipeline* funcione a la perfección y se deben almacenar todos juntos, en la misma carpeta donde se encuentre instalado Nextflow.

La línea de código que se debe introducir para correr el script es: "*./nextflow main.nf*". En el caso de optar por no clonar el repositorio, se puede usar el pipeline desde la terminal empleando dos comandos: "*./nextflow pull https://github.com/gdauria/nf-mtg-engine.git*"; y, posteriormente,"*./nextflow run https://github.com/gdauria/nf-mtg-engine.git -r main --manifest manifest.csv*". El primero 'baja' el repositorio, o lo actualiza si ya se ha bajado previamente, y el segundo corre el *script*.

En el archivo *main.nf* la delimitación entre cada parte (los procesos) se produce mediante la estructura:

process nombre_proceso {
definición_directorio
echo true
definición input:
"""
definición script:
"""
}
Cada proceso se describe detalladamente línea por línea y el código queda representado mediante su limitación con "</>" y el contenido escrito en cursiva, los canales y ficheros se escriben entre comillas, y los comandos entre comillas y en cursiva.

Los parámetros quedan definidos al principio del script tras la línea que implementa Nextflow: *<#!/usr/bin/env nextflow>*

Los procesos y comandos se defines gracias al manual de Nextflow (DOCUMENTATION!, 2020) y de los respectivos manuales de cada paquete de *software* que se emplea en la escritura de los scripts.

3.3. DEFINICIÓN DE PARÁMETROS

El *pipeline* desarrollado para el análisis metagenómico de secuencias empieza recogiendo los parámetros por defecto necesarios para la ejecución de los programas de cada proceso. Estos parámetros se recogen en un fichero de configuración llamado "nextflow.config" que el mismo proceso raíz reconoce y carga de forma automática.

El análisis se lleva a cabo para todas las muestras definidas en el fichero "manifest.csv". Este tiene la estructura de una tabla separada por comas donde cada columna define: "NOMBRE DE

MUESTRA", "SECUENCIAS FORWARD" y "SECUENCIAS REVERSE". En el estado actual el *pipeline* permite solamente el uso de secuencias de Illumina emparejadas. El desarrollo futuro permitirá la elección y el uso de secuencias de otras plataformas.
El fichero de configuración define los siguientes parámetros:

- manifest = "$PWD/manifest.csv" (Fichero de correspondencia entre nombre de muestras y ficheros de secuencias, por defecto el sistema busca uno que se llame "manifest.csv" que se encuentre en la misma carpeta de ejecución del programa);
- bn = "MTG_PROJECT" (Nombre base del proyecto. Por defecto: "MTG_PROJECT");
- THREADS = 12 (Número de procesadores usados por defecto);
- WIN_SIZE = 10 (Tamaño de ventanas deslizantes usadas en el control de calidad);
- MEAN_QUAL = 30 (Calidad media requerida en la ventana deslizante en el proceso del control de calidad);
- MAX_LEN1 = 300 (Longitud máxima de secuencias *Forward*);
- MAX_LEN2 = 300 (Longitud máxima de secuencias *Reverse*);
- CACHEDIR = "$PWD/cache" (Carpeta donde almacenar los programas descargados);
- evaluefun4 = 1e-03
- minidenfun4= 1e-03
- blocksize = 8
- evaluehmmer5 = 1e-10
- assembler = "SPADES" (Ensamblador empleado, por defecto SPADES);
- mode = "CO" (Por defecto se selecciona el modo co-assembly).

Después de haber configurado los parámetros iniciales, el pipeline lee el fichero manifest.csv definiendo los nombres de las muestras a analizar y sus ficheros de secuencias asociados. Estos datos se almacenan en canales específicos que se irán utilizando a lo largo de todos los procesos llamados en este caso: "samples_channel".
A continuación, el pipeline descarga las bases de datos necesarias para el proceso de anotación (ver más adelante).
Empieza entonces la ejecución de los procesos de análisis. Es prudente recordar que Nextflow no sigue un orden secuencial, los proceso se llevarán a cabo nada más tener disponibilidad de los ficheros de entrada y de todos los componentes necesarios.

3.4. PROCESO LIMPIA

El proceso denominado LIMPIA es en el que se produce el control de calidad de las muestras secuenciadas.
El proceso empieza descargando e instalando los programas necesarios desde el entorno conda en la carpeta de cache definida en el fichero de configuración. En concreto va a necesitar del programa fastp. El proceso recibe los inputs desde el canal "samples_channel" que ha leído la tabla manifest.csv.
El paquete de software que realiza esta tarea es fastp. Nextflow crea automáticamente entornos conda, y descarga aquellos paquetes que se especifican mediante el comando "*conda 'canal_conda:: paquete_software'* ". Para obtener fastp se necesita el canal de bioconda (Grüning *et al.*, 2018), quedando la línea de comando de la siguiente forma: *<conda 'bioconda::fastp'>*

Para controlar dónde se dirigen los archivos de salida y enviarlos a una carpeta en concreto existe la herramienta "*publishDir"*. La carpeta creada para los archivos del proceso limpia es "{BASENAME}/cleaned". Existen parámetros opcionales para el comando "*publishDir"*, con los que es posible modificar la creación – por defecto – de un enlace entre cada salida y la carpeta en cuestión. Uno de los parámetros es "*mode*" que tiene varios modos a su vez, para el pipeline el elegido es "*copy*" que copia los archivos de salida en el directorio publicado. Otro parámetro elegido es "*pattern*" que especifica qué archivos *output* se publican en la carpeta de destino de todos los que se producen en el proceso limpia.

<publishDir "${BASENAME}/cleaned", mode: 'copy', pattern: 'cleaned..fastq.gz'*
publishDir "${BASENAME}/cleaned", mode: 'copy', pattern: 'fastp_report.'*
echo true>

Los archivos *input* se proporcionan en formato FASTQ. Dentro de estos archivos cada secuencia tendrá dos lecturas distintas (R1, R2) que corresponderán a la secuenciación de dicha secuencia desde sus dos extremos (*Forward y Reverse*). Ambas lecturas se procesan por separado y entran mediante el canal "samples_channel" definido anteriormente.
<input:
set SampleID, file(R1), file(R2) from samples_channel>

Los ficheros de salida de este proceso serán los archivos FASTQ R1 y R2 "limpios" vertidos a los canales "*R1_CH"* y "R2_CH", respectivamente, para cada secuencia; y los informes de control de calidad en formato json y HTML antes y después del filtrado.
<output:
file "cleaned.${SampleID}.R1.fastq.gz" into R1_ch
file "cleaned.${SampleID}.R2.fastq.gz" into R2_ch
file "fastp_report.">*

En el pipeline, el comando fastp también necesita da algunos parámetros configurados anteriormente por defecto (ver apartado anterior sobre el fichero de configuración "nextflow.config") o definido por el usuario. Finalmente, el comando se aplica de la siguiente forma (Chen *et al*., 2018):
<fastp -i $R1 -I $R2 -o cleaned.${SampleID}.R1.fastq.gz -O cleaned.${SampleID}.R2.fastq.gz --detect_adapter_for_pe --adapter_fasta ${NEXTERAADAPT} --cut_tail --cut_window_size ${WIN_SIZE} --cut_mean_quality ${MEAN_QUAL} --thread ${THREADS} --json fastp_report.json --html fastp_report.html --report_title='fastp_report' > fastp.log>

Los archivos de entrada se definen mediante "-i" (R1) e "-I" (R2); los archivos de salida, mediante "-o" (R1 outfile) y "-O" (R2 outfile). fastp predice adaptadores que pueden llevar las secuencias y los corta secuencia a secuencia mediante "*–detect_adapter_for_pe*", además se le proporciona una biblioteca de adaptadores Nextera, "*—adapter_fasta*". Por otro lado, se recortan las colas 3' de las secuencias, especificado en la línea de script mediante el comando "*cut_window_size*" se marca el tamaño de las ventanas de recorte, y se fija el umbral de calidad mínima con "*—cut_mean_quality*". En caso de que la calidad media de los nucleótidos de la ventana sea inferior al umbral, las bases que la componen se anotarán como descartadas y la ventana se moverá un base adelante para seguir el proceso. Por último, "*--thread*" especifica cuántos subprocesos se acometen a la vez.

3.5. PROCESO CONCAT1 y CONCAT2

Ambos procesos tienen como objetivo aunar en dos archivos diferentes todas las muestras con secuencias forward (CONCAT1) y secuencias reverse (CONCAT2).

El directorio que alberga los archivos de salida es "{BASENAME}/concat" en ambos procesos que convergen en ese punto, mediante el comando *"publishDir"* con los parámetros "mode: 'copy'" y "pattern".

Los archivos de entrada de cada tarea provienen de la recopilación de los elementos que surgen de los canales "R1_ch" y "R2_ch", respectivamente. Gracias al operador ". *collect()*" los elementos emitidos se dirigen a una lista y son devueltos como un único "objeto" para ser tratado por el script.

La definición de entrada del proceso CONCAT1 queda de la siguiente forma:
<input:

file "" from R1_ch.collect()>*
Del mismo modo, el proceso CONCAT2 queda así descrito:
<input:
file "" from R2_ch.collect()>*

El script pretende unir los objetos recibidos como input, que se denotan como "*", mediante el comando "*cat*" en un archivo de salida denominado "concat.R1.fastq.gz" y "concat.R2.fastq.gz" según la naturaleza de las secuencias de entrada (R1 o R2).
script CONCAT1:
*<cat * > concat.R1.fastq.gz>*

script CONCAT2:
*<cat * > concat.R2.fastq.gz>*

Por otro lado, los archivos de salida son enviados a dos canales distintos, uno por proceso, designados como "*R1_assembly_ch*" y "*R2_assembly_ch*".
output CONCAT1:
<output:
file("concat.R1.fastq.gz") into R1_assembly_ch>

output CONCAT2:
<output:
file("concat.R2.fastq.gz") into R2_assembly_ch>

3.6. PROCESO ASSEMBLY

El ensamblaje de las lecturas pulidas en el proceso LIMPIA se trata en el proceso llamado "ASSEMBLY", y el paquete de bioconda empleado es SPAdes.
<conda 'bioconda::spades'>

Los outputs se dirigen a la carpeta "{BASENAME}/assembly" por medio del operador *"publishDir"* con los parámetros "*mode*: *copy*"' y "*pattern*" –igual que en los procesos anteriores– los archivos que se aíslan son los *contigs* creados por el ensamblador.

Los archivos de entrada corresponden a dos ficheros FASTQ, uno de concatenación de lecturas R1 y otro de lecturas R2, procedentes de los canales "R1_assembly_ch" y "R2_assembly_ch" (proceso "CONCAT1" y "CONCAT2").
<input:
file ("concat.R1.fastq.gz") from R1_assembly_ch
file ("concat.R2.fastq.gz") from R2_assembly_ch>

El resultado del ensamblaje (*contigs* obtenidos en formato FASTA) se dirige a dos canales distintos, para su posterior empleo en dos procesos distintos; ya que, no se puede usar el mismo canal en diferentes procesos (*queue channel*).
<output:
file("mtg_assembly/contigs.fasta") into spades_ch1
file("mtg_assembly/contigs.fasta") into spades_ch2>

Los archivos de entrada se definen mediante "*-1*" e "*-2*" y los archivos de salida, mediante "*-o*". SPAdes proporciona diferentes archivos de salida que se guardan en el directorio elegido por el usuario. Por lo tanto, en la carpeta temporal – cada proceso crea la suya – se almacenan los siguientes una gran variedad de archivos, si bien es cierto que para el pipeline solo interesa el siguiente que llegará a ambos canales (Prjibelski *et al*., 2014, Vasilinetc *et al*., 2015 y Nurk *et al*., 2017.):

- “nombre_directorio/contigs.fasta”: archivo que contiene los *contigs* resultantes del ensamblaje.

3.7. PROCESO PROKKA

Este proceso lleva a cabo a la anotación estructural de los *contigs* que provienen del ensamblado.
<conda "${projectDir}/envs/prokka.yml">

En este caso para el software necesario (Prokka) se descargará automáticamente e instalará en un nuevo ambiente conda mediante un archivo de instalación que define los canales y paquetes necesarios. La versión de Prokka elegida es 1.14.6. por su más amplia compatibilidad con distintos sistemas operativos y sus versiones.
El fichero input es el archivo que contiene los *contigs* ensamblados y proceden del canal “*spades_ch1*”.
<input:
file ("contigs.fasta") from spades_ch1>

Prokka devuelve varios ficheros de salida que se detallaron anteriormente en la Tabla 3. En concreto, para los pasos a seguir se necesitará de los archivos “.faa” y “.gff”. En el primero aparecen las proteínas anotadas en formato FASTA, el segundo es un archivo GFF v3 que contiene las secuencias y anotaciones en un formato tabular estándar. En este proceso se crean cuatro canales, tres de ellos (“prokka_ch_1, prokka_ch_2 y prokka_ch_4*”*) contienen el archivo con extensión “.faa”, y el canal “prokka_ch_3*”*, el archivo “.gff” (Seemann, 2014).
<output:
file("prokka_out/${BASENAME}.faa") into prokka_ch_1
file("prokka_out/${BASENAME}.faa") into prokka_ch_2
file("prokka_out/${BASENAME}.gff") into prokka_ch_3
file("prokka_out/${BASENAME}.faa") into prokka_ch_4
file("prokka_out/")>*

En el script, el *input* se precisa escribiendo el archivo después del comando “*prokka*”, el directorio al que se dirige el output mediante “*—outdir*”, y “*-cpus ${THREADS}”* precisa el número de procesos que van a discurrir a la vez.

En la versión de Nextflow empleada en esta tesis, los canales creados no pueden repetirse así que cuando necesitamos del mismo fichero para varios procesos, esto se tiene que repetir tantas veces cuantas veces se necesite. Por esta razón, vamos a necesitar de tres canales distintos para redirigir el fichero “BASENAME.faa” para la anotación funcional donde se emplean tres procesos distintos de anotación, dos paquetes de software – DIAMOND y HMMER – y tres bases de datos – KEGG, EGNOG y Pfam –. Estos procedimientos se efectúan en paralelo, puesto que el archivo de entrada es siempre el mismo y procedente del proceso “PROKKA”.

El directorio creado para almacenar los resultados de los tres procesos de anotación es "{BASENAME}/annotation".

3.8. PROCESO annotationKegg

El ambiente conda necesario para este proceso se detalla en el archivo “diamond.yml” (ver Anexo I). La versión de DIAMOND implementada es 0.9.22 y se implementa también Python 2.7, necesario por problemas de implementación de DIAMOND con versiones posteriores.
<conda "${projectDir}/envs/diamond.yml">

Los archivos de entrada cuando se realiza la anotación funcional son, además del fichero con las secuencias de proteínas a analizar, la base de datos contra la que se produce la anotación. La

base de datos KEGG ("keggdb.dmnd") se integra en el script gracias al proceso getDB. Esta descarga automáticamente dicha base de datos, proporcionada para la tesis por FISABIO (Fundación para el Fomento de la Investigación Sanitaria y Biomédica de la Comunidad Valenciana). El archivo requerido es uno de los tres con extensión ".faa" – designado como FAA – desde el canal 1 que sale del proceso PROKKA.
<input:
file "FAA" from prokka_ch_1
file "keggdb.dmnd" from keggdb_ch>

En el *script* se distinguen una serie de comandos (Buchfink *et al*., 2021):

- "*-q*": define el archivo objeto del ensamblaje.
- "*-d*": define la base de datos empleada.
- "*-e*/*—evalue":* especifica el valor máximo esperado para que considerar un alineamiento exitoso. Por defecto el valor sería 0,001.
- "*—id*", comunica alineamientos que estén por encima del porcentaje de identidad de secuencia que proporciona el parámetro.
- "*–block-size*/*-b*" que indica el tamaño de los bloques de caracteres (letras) de secuencia (en billones) que se van a procesar a la vez. Este parámetro es el más importante en el control del uso de la memoria del programa. Por defecto el valor del uso de la memoria es 12 GB.

Los parámetros que se le dan a los comandos son los siguientes – que ya se han explicado con anterioridad –:

- FAA = archivo de entrada (".faa"): parámetro perteneciente al comando "*-q*".
- keggdb.dmnd = base de datos KEGG: parámetro perteneciente al comando "*-d*".
- evaluefun4 = $1e^{-03}$: parámetro perteneciente al comando "*—evalue*/*-e*".
- minidenfun4 = $1e^{-03}$: párametro perteneciente al comando "*—id*".
- blocksize = 8: parámetro perteneciente al comando "*–block-size*/*-b*".

<diamond blastp -q ${FAA} -p ${THREADS} -d ${KEGGDB} -e $evaluefun4 --id $minidenfun4 --quiet -b $blocksize -f 6 qseqid qlen sseqid slen pident length evalue bitscore qstart qend sstart send -o annotation.kegg.diamond>

El archivo de salida se nombra como "annotation.kegg.diamond" y se deriva al canal "*kegg_channel2*".
<output:
file ("annotation.kegg.diamond") into kegg_channel>

3.9. PROCESO annotationCog

Como su nombre indica, este proceso es muy similar al annotationKegg porque el paquete de software para anotar es el mismo (DIAMOND versión 0.9.22) descrito por "diamond.yml".
<conda "${projectDir}/envs/diamond.yml">

El *input* es, de nuevo, el archivo que contiene las proteínas ensambladas (con extensión ".faa"), esta vez del canal "prokka_ch_2" y la base de datos empleada para la anotación se descarga automáticamente gracias al proceso getiEggnogDb, procedente de los recursos de FISABIO, y entra como input desde el canal "eggnogdb_ch".
<input:
file "eggnog.dmnd" from eggnogdb_ch
file "FAA" from prokka_ch_2>
El output se denomina "annotation.cog.diamond" y el canal de salida, "cog_channel".
<output:
file ("annotation.cog.diamond") into cog_channel>

Por último, el script alberga los mismos comandos y parámetros que en el proceso referente a la base de datos KEGG; exceptuando la base de datos, que en este caso corresponde a la base EGGNOG.
<diamond blastp -q ${FAA} -p ${THREADS} -d ${DBFOLDER}/eggnog.dmnd -e $evaluefun4 --id $minidenfun4 --quiet -b $blocksize -f 6 qseqid qlen sseqid slen pident length evalue bitscore qstart qend sstart send -o annotation.cog.diamond>

3.10. PROCESO annotationPfam

El tercer proceso de anotación se lleva a cabo mediante un paquete de software distinto, HMMER.
<input:
conda 'bioconda::hmmer'>

No obstante, el archivo de entrar es el mismo que en las dos anotaciones anteriores, en esta ocasión del canal "prokka_ch_4". La base de datos Pfam se recoge gracias a los canales "pfamdb_ch" y "pfamdatdb_ch" procedentes del proceso getPfamADb.
<input:
file "FAA" from prokka_ch_4
file "Pfam-A.hmm" from pfamdb_ch
file "Pfam-A.hmm.dat" from pfamdatdb_ch>

El archivo output se designa como "annotation.pfam.hmm".
<output:
file ("annotation.pfam.hmm")>

Como el programa de anotación ha cambiado, los comandos y parámetros también lo han hecho (Eddy, 2020):

- "*—cpu*": define el número de hilos <n> que se ejecutan en paralelo. En realidad, existe un hilo maestro, por lo que el número real de hilos corresponde a <n> +1.
- "*—domtblout*": marca la salida y almacenaje de un archivo delimitado por espacios que recoja el resumen de los datos de salida, dividiéndolos por dominios. Debe aparecer una línea de datos por cada dominio homólogo detectado en la secuencia de consulta por cada modelo de homología.
- "*-E*": informa de los falsos positivos por consulta, los perfiles con un *e-value* <= <x>. Por defecto, el valor es 10; el programa informa de unos 10 falsos positivos por consulta. Este comando ayuda a reducir el ruido de fondo.

Los parámetros que se le dan a los comandos son los siguientes:

- evaluehmmer5: 1e-10: parámetro perteneciente al comando "*-E*".
- THREADS: 12: parámetro perteneciente al comando "*—cpu*".

<hmmsearch --domtblout annotation.pfam.hmm -E ${evaluehmmer5} --cpu ${THREADS} ${DBFOLDER}/Pfam-A.hmm ${FAA}>

3.11. PROCESO MAPPING

El proceso de mapeo se indica como MAPPING. El mapeo se produce sobre el archivo "mtg_assemblymapping" perteneciente al proceso de ensamblaje y las secuencias originales (genoma de referencia). Los programas requeridos para este proceso son BWA y SAMtools, este último en versión 1.9.
<conda 'bioconda::bwa bioconda::samtools=1.9'>

El genoma de referencia proviene de proceso de ensamblaje con SPAdes (canal 2).

<input:
file("mtg_assemblymapping") from spades_ch2
set SampleID, R1, R2 from samples_channel_2>

Los archivos de salida son un fichero por cada muestra que lleva el nombre de dicha muestra y la extensión ".bam" que se derivan en su totalidad al canal "mappingResult".
<output:
file("${SampleID}.bam")
set SampleID, file("${SampleID}.bam") into mappingResult>

El primer paso es indexar el genoma de referencia produciendo para los programas sigueintes, tanto el bwa como para samtools; para ello, se usan los comandos *"bwa index {mtg_assemblymapping}" y "samtools faidx {mtg_assemblymapping}"* (Li, 2021a).
<bwa index ${mtg_assemblymapping}
samtools faidx ${mtg_assemblymapping}>

Por otro lado, el comando para el mapeo en el *script* es "*bwa mem"* que tiene diversas opciones:

- "*-Y*".
- "*-M*": corresponde al marcaje, como secundarias, las secuencias divididas más cortas y así forzar la compatibilidad con Picard.
- *"-R*": se emplea para especificar la línea de cabecera que será la etiqueta en el archivo de salida SAM.

<bwa mem -Y -M -R "@RG\\tID:${SampleID}\\tLB:${SampleID}\\tSM:${SampleID}" -t ${THREADS} ${mtg_assemblymapping} $R1 $R2 | samtools view -Sb - | samtools sort > ${SampleID}.bam>

Posterior al tratamiento con BWA (alineamiento) se procesa el resultado mediante el programa SAMtools (Li, 2021b):
<| samtools view -Sb - | samtools sort > ${SampleID}.bam>

Los comandos elegidos son "*view"* y "*sort"*:

- *Samtools view*: el programa interpreta automáticamente el tipo de datos en entrada (*-S*) y dará como salida el mismo alineamientos en formato BAM (*-b*).

- *Samtools sort*: ordena las alineaciones por las coordenadas más a la izquierda. Este comando añade una etiqueta de cabecera de ordenación apropiada. Con el comando del script se especifica que el archivo output "ordenado" se escribe en el fichero final de salida "SampleID.bam"

Por acabar con el proceso, el archivo de salida ordenado va a ser indexado con "*samtools index*", se debe utilizar la ordenación por coordenadas por defecto.

3.12. PROCESO HTSEQ

El último proceso descrito en el pipeline es el referente al análisis de expresión diferencial, que en este caso usaremos para calcular la cobertura de cada metagenoma sobre los genes anotados de cada ensamblado.
Los paquetes de software empleados para el tratamiento de los datos son "htseq" y "pysam", obtenidos gracias a un ambiente conda:
<conda 'bioconda::htseq bioconda::pysam'>

Los inputs provienen de dos vertientes distintas. Por un lado, los archivos ".bam" correspondientes al mapeo de cada muestra y que entran mediante el canal "*mappingResult*", y

un archivo que contiene la anotación almacenada en el canal "*prokka_ch_3*" con extensión "gff".
<input:
set SampleID, bamFile from mappingResult
file annotation from prokka_ch_3>

El fichero de salida se guarda en el canal "*htseq_out_channel*" con extensión ".htseq"
<output:
file ".htseq" into htseq_out_channel>*
Por ultimo, el script se compone de dos partes:
La primera línea tiene la función de eliminar la última línea de un fichero.
<sed '/##FASTA/,\$d' $annotation > ${annotation}.correct>

El resto de las líneas corresponden al grueso del script correspondiente al paquete "*htseq-count"*, quedando de esta manera (Anders *et al*., 2015):
<htseq-count -t "CDS" -f "bam" -i "locus_tag" --additional-attr="inference" --additional-attr="product" --additional-attr="gene" --additional-attr="Name" --stranded=no $bamFile ${annotation}.correct > ${SampleID}.htseq>

- "*-t*": tipo de característica que se va a utilizar, el resto se ignoran en el análisis. (3º columna del GFF file). El parámetro del comando es CDS.

- "*-f*": describe el formato del input, por defecto es SAM. El valor adjudicado a este parámetro es "bam" (archivo BAM).

- "*-i*": Atributo GFF que se utilizará como ID de la característica, en este caso el atributo es "locus tag". Varias líneas GFF con el mismo ID se considerarán partes de la misma característica. El ID de característica se utiliza para identificar los recuentos en la tabla de salida.

El resto de las características se marcan como adicionales.

- *Stranded*: si los datos son de una hebra específica.
  En el caso de stranded=no, se considera que una lectura se solapa con un rasgo, independientemente de que se mapee en la misma hebra o en la opuesta al rasgo.

Todos los procesos anteriormente expuestos son los que componen el análisis metagenómico descrito en *main.nf* y se exponen visualmente. – a modo de resumen-esquema – en la Figura 6.

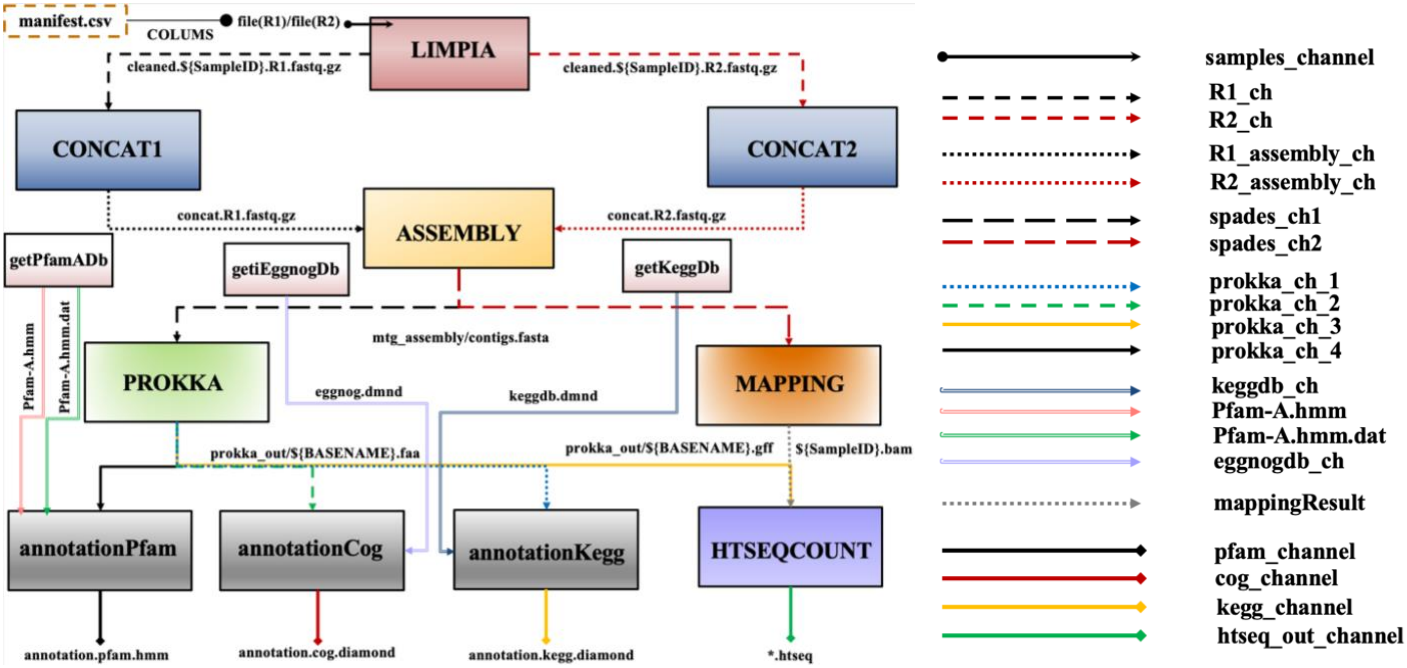

Figura 6. Esquema de *main.nf*. Los canales que conectan los procesos se detallan en la leyenda de la derecha. El primer archivo input del *script* es manifest.csv que entra en el proceso LIMPIA, las flechas representan los canales que albergan los archivos output y los conducen al siguiente análisis. Los archivos de salida finales son "annotation.pfam.hmm", "annotation.cog.diamond", "annotation.kegg.diamond" y "*.htseq".

4. BENCHMARKING

Para el *benchmarking* del *pipeline* se ha empleado un *dataset* procedente de una muestra metagenómica descargada desde la base de datos EBI MGnify (https://www.ebi.ac.uk/metagenomics), estudio: MGYS00001559, muestra: ERS1571320. Se ha elegido una sola muestra para intentar garantizar un ambiente homogéneo de lo que obtener el *dataset* de prueba. Este *dataset* inicial se ha repartido en tres ficheros de 250.000 secuencias cada uno. Estos ficheros se han identificado con los nombres de muestra1, muestra2 y muestra3 y se han incluído en el fichero manifest.csv para comprobar la duración del análisis mediante el pipeline basado en Nextflow, objeto de esta tesis, contra el procesamiento secuencial de las tres muestras.

El proceso se ha llevado a cabo en un servidor de cálculo ASUS con 64 hilos y 256Gb de memoria RAM. Para este *benchmarking* se han usado solo 6 hilos para limitar los recursos mientras que la memoria RAM no ha tenido ninguna limitación de uso sin llegar nunca a saturación, siendo el proceso annotationCog el más demandante. El uso de la CPU cuantifica cómo se comparte el procesador entre los programas del script, siendo los procesos de annotationKegg y annotationCog los que más CPU "emplean". Tanto el uso de la memoria RAM como el de la CPU se muestra en la Figura 7:

- Tiempo total de cálculo usando Nextflow: 00h 48m 18s. La cronología de la ejecución de los procesos muestra que los procesos ASSEMBLY y PROKKA son los que más tiempo engloban (Figura 8), ya que el ensamblaje y la anotación estructural son los procesos que más recursos emplean y más exhaustivo es su abordaje.
- Tiempo total de cálculo secuencial: 2h 19m 28s.

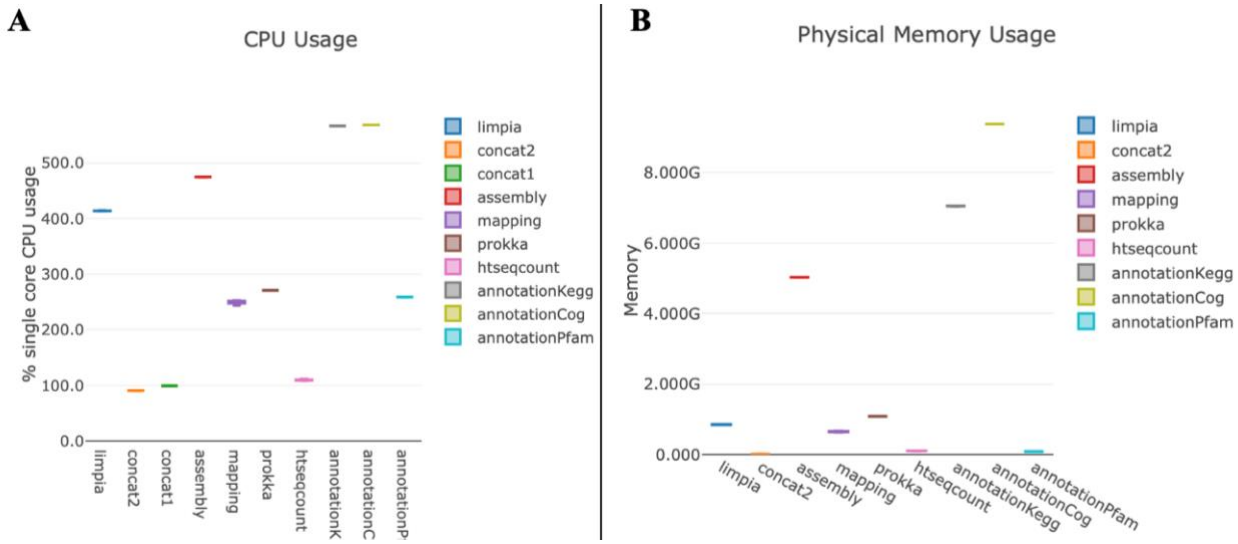

Figura 7. A) Gráfico del uso de la CPU en cada proceso en *main.nf*. B) Gráfico del uso de la memoria RAM de cada proceso en *main.nf*.

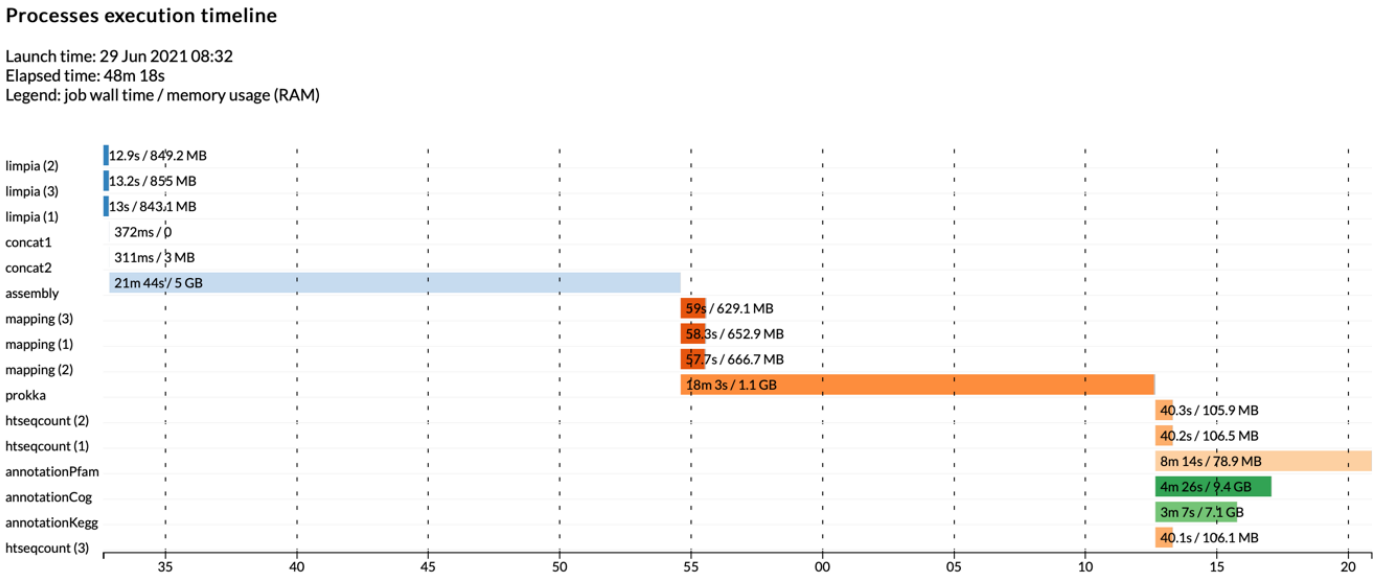

Figura 8. Cronología de la ejecución de los procesos de *main.nf*. Los números que aparecen entre paréntesis al lado de cada proceso corresponden con la muestra que se procesa en cada caso, existiendo tres muestras distintas en "manifest.csv" para el benchmarking.

5. CONCLUSIONES

El objetivo del presente trabajo era explorar las posibilidades de Nexflow como lenguaje de programación idóneo para la creación de *pipelines* de análisis de datos metagenómicos. Las características principales que ofrece Nextflow y que resuelven las dificultades que derivan de este tipo de análisis son las siguientes:

- Capacidad de ser implementado en una amplia gama de lenguajes de programación.

- Ejecución con recursos computacionales medios.
- Paralelización de los procesos integrados en el pipeline.
- Uso de contenedores multiescala para la creación de entornos.
- Integración en repositorios de software públicos.
- Aprovechamiento del almacenamiento interno.
- Registro de la evolución en la ejecución del *pipeline*.

Todas ellas tienen potencial para cubrir las necesidades planteadas por las nuevas tecnologías de secuenciación y las particularidades de los datos metagenómicos.
En primer lugar, la implementación en cualquier lenguaje hace más accesible su empleo a usuarios con conocimientos en lenguajes básicos. El hecho de que no sean necesarios recursos computacionales de especializados facilita el análisis de datos en cualquier lugar con conexión a Internet. Asimismo, la paralelización agiliza los análisis – muy arduos y complicados a medida que se añaden procesos– que por separado son considerablemente más lentos que cuando son integrados en un pipeline implementado en Nextflow.

En cuanto a las perspectivas de futuro, se ha desarrollado una extensión de Nextflow denominada DSL2. DSL2 pretende separar las dos partes básicas en las que se puede dividir un *pipeline:* la parte en la que cada proceso o tarea intenta correr el *script* y la parte que intenta conectar todos los procesos. Las mejoras que se implementan gracias a ello son principalmente la modularización de los *pipelines*, re-utilización de componentes y agilización de la definición de patrones de aplicación recurrentes. En el caso de Nextflow, los *pipelines* se entienden como un bloque indivisible por la definición de *inputs* y *outputs* que unen cada pieza de software de forma monolítica. En la nueva versión no es necesario "declarar" el canal de donde provienen los archivos, quedándose obsoletos los operadores "into" y "to".
DSL2 es una "extensión natural" de DSL1, por lo que no implica una excesiva dificultad añadida para los usuarios que ya han usado Nexflow (NF-CORE, 2020, 3m 37s).

Con todo esto, Nextflow y sus futuras implementaciones se postulan como respuesta ante las nuevas capacidades de las NGS que traen consigo la necesidad de analizar grandes cantidades de muestras, en muchas ocasiones, *in situ* con recursos computacionales limitados y en un tiempo reducido.

El pipeline objeto de esta tesis se está ahora actualizando a la versión DSL2 de Nextflow, aumentado la robustez y velocidad de ejecución. El protocolo aquí presentado se encarga de llevar a cabo los pasos más complejos, desde un punto de vista computacional, del análisis metagenómico. El desarrollo futuro de este *pipeline* va a incluir la posibilidad de elección de distintos ensambladores o protocolos de ensamblados secuenciales o por muestra, a elegir por el usuario. Además, se está acoplando la salida de datos a otras herramientas como SqueezeMeta para que, eventualmente, el mismo pipeline pueda integrarse en este flujo de análisis.

6. BIBLIOGRAFÍA

[Data Processing of Nextera Mate Pair Reads on Illumina Sequencing Platforms] (17 de diciembre de 2012).

ABUBUCKER, S.; SEGATA, N.; GOLL, J.; SCHUBERT, A. M.; IZARD, J.; CANTAREL, B. L.; ZUCKER, J.; THIAGARAJAN, M.; HENRISSAT, B.; WHITE, O., KELLEY, S.; METHÉ, B.; SCHLOSS, P.; GEVERS, D.; MITREVA, M.; HUTTENHOWER, C. (2012). Metabolic reconstruction for metagenomic data and its application to the human microbiome. *PLoS Computational Biology*, 8(6), e1002358.

ADAMS, M. D.; KELLEY, J. M.; GOCAYNE, J. D.; DUBNICK, M.; POLYMEROPOULOS, M. H.; XIAO, H.; MERRIL, C. R.; WU, A.; OLDE, B.; MORENO, R. F.; KERLAVAGE, A. R.; MCCOMBIE, W. R.; VENTER, J. C.M(1991). Complementary

DNA sequencing: expressed sequence tags and human genome project. *Science* (New York, N.Y.), 252(5013), 1651-1656.

AL-OKAILY, A. A. (2016). HGA: de novo genome assembly method for bacterial genomes using high coverage short sequencing reads. *BMC Genomics*, 17(1), 193.

ALTSCHUL, S. F.; GISH, W.; MILLER, W.; MYERS, E. W.; LIPMAN, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215(3), 403–410.

ALTSCHUL, S. F.; MADDEN, T. L.; SCHÄFFER, A. A.; ZHANG, J.; ZHANG, Z.; MILLER, W.; LIPMAN, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17), 3389–3402.

AMBARDAR, S.; GUPTA, R.; TRAKROO, D.; LAL, R.; VAKHLU, J. (2016). High throughput sequencing: An overview of sequencing chemistry. *Indian journal of microbiology*, 56(4), 394–404.

ANDERS, S.; PYL, P. T.; HUBER, W. (2015). HTSeq--a Python framework to work with high-throughput sequencing data. *Bioinformatics* (Oxford, England), 31(2), 166–169.

ARUMUGAM, M.; HARRINGTON, E. D.; FOERSTNER, K. U.; RAES, J.; BORK, P. (2010). SmashCommunity: a metagenomic annotation and analysis tool. *Bioinformatics* (Oxford, England), 26(23), 2977–2978.

AZIZ, R. K.; BARTELS, D.; BEST, A. A.; DEJONGH, M.; DISZ, T.; EDWARDS, R. A.; FORMSMA, K.; GERDES, S.; GLASS, E. M.; KUBAL, M.; MEYER, F.; OLSEN, G. J.; OLSON, R.; OSTERMAN, A. L.; OVERBEEK R. A.; MCNEIL, L. K.; PAARMANN, D.; PACZIAN, T.; PARRELLO, B.; PUSCH, G. D.; REICH, C.; STEVENS, R.; VASSIEVA, O.; VONSTEIN, V.; WILKE, A.; ZAGNITKO, O. (2008). The RAST Server: rapid annotations using subsystems technology. *BMC Genomics*, 9, 75.

BANKEVICH, A.; NURK, S.; ANTIPOV, D.; GUREVICH, A. A.; DVORKIN, M.; KULIKOV, A. S.; LESIN, V. M.; NIKOKENKO, S. I.; PHAM, S.; PRJIBELSKI, A. D.; PYSHKIN, A. V.; SIROTKIN, A. V.; VYAHHI, N.; TESLER, G.; ALEKSEYEV, M. A.; PEVZNER, P. A. (2012). SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*: A Journal of Computational Molecular Cell Biology, 19(5), 455–477.

BEN-KIKI, O.; EVANS, C.; DÖT NET, I. (2009).*Yaml ain't markup language (YAMLTM)* (1.2.) https://yaml.org/spec/1.2/spec.html.

BENSON, D. A.; KARSCH-MIZRACHI, I.; LIPMAN, D. J.; OSTELL, J.; WHEELER, D. L. (2005). *GenBank.* Nucleic Acids Research, 33(Database issue), 34-38.

BESEMER, J.; LOMSADZE, A.; BORODOVSKY, M. (2001). GeneMarkS: a self-training method for prediction of gene starts in microbial genomes. Implications for finding sequence motifs in regulatory regions. *Nucleic Acids Research*, 29(12), 2607–2618.

BOISVERT, S.; RAYMOND, F.; GODZARIDIS, E.; LAVIOLETTE, F.; CORBEIL, J. (2012). Ray Meta: scalable de novo metagenome assembly and profiling. *Genome Biology*, 13(12), 122.

BRUIJN, DE, N. G. (1946). *A combinatorial pro*blem. Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam, 49(7), 758-764.

BUCHFINK, B.; REUTER, K.; DROST, H.-G. (2021). Sensitive protein alignments at tree-of-life scale using DIAMOND. *Nature Methods*, 18(4), 366–368.

BUCHFINK, B.; XIE, C.; HUSON, D. H. (2015). Fast and sensitive protein alignment using DIAMOND. *Nature Methods*, 12(1), 59–60.

CHAUDHURI, R. R.; LOMAN, N. J.; SNYDER, L. A. S.; BAILEY, C. M.; STEKEL, D. J.; PALLEN, M. J. (2008). xBASE2: a comprehensive resource for comparative bacterial genomics. *Nucleic Acids Research*, 36(Database issue), 543-546.

CHEN, S.; ZHOU, Y.; CHEN, Y.; GU, J. (2018). fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics* (Oxford, England), 34(17), 884–890.

CHEVREUX, B.; PFISTERER, T.; DRESCHER, B.; DRIESEL, A. J.; MÜLLER, W. E. G.; WETTER, T.; SUHAI, S. (2004). Using the miraEST assembler for reliable and automated mRNA transcript assembly and SNP detection in sequenced ESTs. *Genome Research*, 14(6), 1147–1159.

CHITSAZ, H.; YEE-GREENBAUM, J. L.; TESLER, G.; LOMBARDO, M.-J.; DUPONT, C. L., BADGER, J. H.; NOVOTNY, M.; RUSCH, D. B.; FRASER, L. J.; GORMLEY, N. A.; SCHULZ-TRIEGLAFF, O.; SMITH, G. P.; EVERS, D. J.; PEVZNER, P. A.; LASKEN, R. S. (2011). Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. *Nature Biotechnology*, 29(10), 915–921.

CHU, Y.; COREY, D. R. (2012). RNA sequencing: platform selection, experimental design, and data interpretation. *Nucleic Acid Therapeutics*, 22(4), 271-274.

CLARKE, J.; WU, H.-C.; JAYASINGHE, L.; PATEL, A., REID, S.; BAYLEY, H. (2009). Continuous base identification for single-molecule nanopore DNA sequencing. *Nature Nanotechnology*, 4(4), 265–270.

COCK, P. J. A.; FIELDS, C. J.; GOTO, N., HEUER, M. L.; RICE, P. M. (2010). The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6), 1767–1771.

COMPEAU, P. E. C; PEVZNER, P. A.; TESLER, G. (2011). How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11), 987-991.

CRG-EU. (s.f.). El Formato GTF: https://bioinformaticaupf.crg.eu/2005/projectes05/3.8/gtf.html.

DERISI, J. L.; IYER, V. R.; BROWN, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science* (New York, N.Y.), 278(5338), 680–686.

DI TOMMASO, P.; CHATZOU, M.; FLODEN, E. W.; BARJA, P. P.; PALUMBO, E.; NOTREDAME, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4), 316–319.

DYMARSKI, P. (Ed.). (2011). Hidden Markov models, theory and applications. InTech.

EBERWINE, J.; SUL, J.-Y.; BARTFAI, T.; KIM, J. (2014). The promise of single-cell sequencing. *Nature Methods*, 11(1), 25–27.

ECSEQ BIOINFORMATICS (20 de marzo de 2020). *What is mate pair sequencing for?* https://www.ecseq.com/support/ngs/what-is-mate-pair-sequencing-useful-for.

EDDY, S. R. (1998). Profile hidden Markov models. *Bioinformatics* (Oxford, England), 14(9), 755–763.

EDDY, S. R. (2020) *HMMER User's Guide Biological sequence analysis using profile hidden Markov models* (3.3.2). Maryland: National Human Genome Research Institute.: http://eddylab.org/software/hmmer/Userguide.pdf.

EID, J.; FEHR, A.; GRAY, J.; LUONG, K.; LYLE, J.; OTTO, G.; PELUSO, P.; RANK, D.; BAYBAYAN, P.; BETTMAN, B.; BIBILLO, A.; BJORNSON, K.; CHAUDHURI, B.; CHRISTIANS, F.; CICERO, R.; CLARK, S.; DALAL, R.; DEWINTER, A.; DIXON, J.; FOQUET, M.; GAERTNER, A.; HARDENBOL, P.; HEINER, C.; HESTER, K.; HOLDEN, D.; KEARNS, G.; KONG, X.; KUSE, R.; LACROIX, Y.; LIN, S.; LUNDQUIST, P.; MA, C.; MARKS, P.; MAXHAM, M.; MURPHY, D.; PARK, I.; PHAM, T.; PHILLIPS, M.; ROY, J.; SEBRA, R.; SHEN, G.; SORENSON, J.; TOMANEY, A.; TRAVERS, K.; TRULSON, M.; VIECELI, J.; WEGENER, J.; WU, D.; YANG, A.; ZACCARIN, D.; ZHAO, P.; ZHONG, F.; KORLACH, J.; TURNER, S. (2009). Real-time DNA sequencing from single polymerase molecules. *Science* (New York, N.Y.), 323(5910), 133–138.

ENSEMBL. (s.f.) *GFF/GTF File Format - Definition and supported options*. http://m.ensembl.org/info/website/upload/gff.html.

EREN, A. M.; ESEN, Ö. C.; QUINCE, C.; VINEIS, J. H.; MORRISON, H. G.; SOGIN, M. L.; DELMONT, T. O. (2015). Anvi'o: an advanced analysis and visualization platform for 'omics data. *PeerJ*, 3(e1319), e1319.

FERRAGINA, P.; MANZINI, G. (2005). Indexing compressed text. *Journal of the ACM*, 52(4), 552-581.

FINN, R. D. (2006). Pfam: clans, web tools and services. *Nucleic acids research*, 34(90001), 247–251.

FINN, ROBERT D.; BATEMAN, A.; CLEMENTS, J.; COGGILL, P.; EBERHARDT, R. Y.; EDDY, S. R.; HEGER, A.; HETHERINGTON, K.; HOLM, L.; MISTRY, J.;

SONNHAMMER, E. L. L.; TATE, J.; PUNTA, M. (2014). Pfam: the protein families database. *Nucleic Acids Research*, 42(Database issue), 222-230.

FINN, ROBERT D.; MISTRY, J.; TATE, J.; COGGILL, P.; HEGER, A.; POLLINGTON, J. E.; GAVIN, O. L.; GUNASEKARAN, P.; CERIC, G.; FORSLUND, K.; HOLM, L.; SONNHAMMER, E. L. L.; EDDY, S. R.; BATEMAN, A. (2010). The Pfam protein families database. *Nucleic acids research*, 38(suppl_1), 211–222.

FONSECA, N. A.; MARIONI, J.; BRAZMA, A. (2014). RNA-Seq gene profiling--a systematic empirical comparison. *PloS One*, 9(9), e107026.

GARIJO, D.; KINNINGS, S.; XIE, L.; XIE, L.; ZHANG, Y.; BOURNE, P. E.; GIL, Y. (2013). Quantifying reproducibility in computational biology: The case of the tuberculosis drugome. *PloS One*, 8(11), e80278.

GRÜNING, B.; THE BIOCONDA TEAM; DALE, R.; SJÖDIN, A.; CHAPMAN, B. A.; ROWE, J.; TOMKINS-TINCH, C. H.; VALIERIS, R.; KÖSTER, J. (2018). Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nature Methods*, 15(7), 475–476.

HANDELSMAN, J.; RONDON, M. R.; BRADY, S. F.; CLARDY, J.; GOODMAN, R. M. (1998). Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chemistry & Biology*, 5(10), 245-249.

HANDELSMAN, J.; LILES, M.; MANN, D.; RIESENFELD, C.; GOODMAN, R. M. (2002). Cloning the metagenome: Culture-independent access to thediversity and functions of the uncultivated microbial world. *Methods in Microbiology*, 241–255.

HILLIS, D. M.; HELLER, H. C.; HACKER, S. D.; HALL, D. W.; LASKOWSKI, M. J.; SADAVA, D. E. (2020). *Life: The science of biology* (12a ed.). 12a ed.

HOFF, K. J.; LINGNER, T.; MEINICKE, P.; TECH, M. (2009). Orphelia: predicting genes in metagenomic sequencing reads. *Nucleic Acids Research*, 37(Web Server issue), W101-5.

HUERTA-CEPAS, J.; SZKLARCZYK, D.; HELLER, D.; HERNÁNDEZ-PLAZA, A.; FORSLUND, S. K.; COOK, H.; MENDE, D. R.; LETUNIC, I.; RATTEI, T.; JENSEN, L. J.; VON MERING, C.; BORK, P. (2019). eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. *Nucleic Acids Research*, 47(D1), 309–314.

HYATT, D.; CHEN, G.-L.; LOCASCIO, P. F.; LAND, M. L.; LARIMER, F. W.; HAUSER, L. J. (2010). Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics*, 11(1), 119.

ILLUMINA. (3 de octubre de 2020) *FASTQ files explained*. https://emea.support.illumina.com/bulletins/2016/04/fastq-files-explained.html.

ILLUMINA. (2020). *Advantages of paired-end and single-read sequencing*. https://emea.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html.

ILLUMINA. (s.f.). *RNA-Seq Alignment v1.0 Online Help. BAM File Format.* https://support.illumina.com/help/BS_App_RNASeq_Alignment_OLH_1000000006112/Content/Source/Informatics/BAM-Format.htm.

ILLUMINA. (s.f.) *Mate pair sequencing*. https://emea.illumina.com/science/technology/next-generation-sequencing/mate-pair-sequencing.html.

ILLUMINA. (s.f.) *Secuenciación: tecnología de Illumina* [Vídeo]. Disponible en: https://support.illumina.com/content/dam/illumina-support/courses/sequencing-illumina-technology-wbt-esp/story_html5.html?iframe.

JENSEN, L. J.; JULIEN, P.; KUHN, M.; VON MERING, C.; MULLER, J.; DOERKS, T.; BORK, P. (2008). eggNOG: automated construction and annotation of orthologous groups of genes. *Nucleic Acids Research*, 36(Database issue), 250-254.

JIANG, H.; WONG, W. H. (2008). SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics* (Oxford, England), 24(20), 2395–2396.

KANEHISA, M.; GOTO, S.; KAWASHIMA, S.; OKUNO, Y.; HATTORI, M. (2004). The KEGG resource for deciphering the genome. *Nucleic Acids Research*, 32(Database issue), 277-280.

KIM, J.; KIM, M. S.; KOH, A. Y.; XIE, Y.; ZHAN, X. (2016). FMAP: Functional Mapping and Analysis Pipeline for metagenomics and metatranscriptomics studies. *BMC Bioinformatics*, 17(1), 420.

KIRCHNER, R. (2018) *Tools for Processing UMI RNA-Tag Data*. GITHUB. https://github.com/vals/umis.

KOREN, S.; TREANGEN, T. J.; POP, M. (2011). Bambus 2: scaffolding metagenomes. *Bioinformatics* (Oxford, England), 27(21), 2964–2971.

LANGMEAD, B.; TRAPNELL, C.; POP, M.; SALZBERG, S. L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3), 25.

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* (Oxford, England), 31(10), 1674–1676.

LI, H. (2012) *bwa - Burrows-Wheeler Alignment Tool* (Versión 0.7.1). Cambridge: Sanger Institute. http://bio-bwa.sourceforge.net/bwa.shtml.

LI, H. (2021). *samtools index – indexes SAM/BAM/CRAM files* (1.12). Cambridge: Sanger Institute. http://www.htslib.org/doc/samtools-index.html.

LI, H. (2021). *samtools index – sorts SAM/BAM/CRAM files* (1.12). Cambridge: Sanger Institute. http://www.htslib.org/doc/samtools-sort.html.

LI, H.; DURBIN, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* (Oxford, England), 25(14), 1754–1760.

LI, H.; HANDSAKER, B.; WYSOKER, A.; FENNELL, T.; RUAN, J.; HOMER, N.; MARTH, G.; ABECASIS, G.; DURBIN, R.; 1000 GENOME PROJECT DATA PROCESSING SUBGROUP. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics* (Oxford, England), 25(16), 2078–2079.

LI, H.; HOMER, N. (2010). A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5), 473–483.

LI, H.; RUAN, J.; DURBIN, R. (2008). Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11), 1851–1858.

LI, R.; LI, Y.; KRISTIANSEN, K.; WANG, J. (2008). SOAP: short oligonucleotide alignment program. *Bioinformatics* (Oxford, England), 24(5), 713–714.

LI, R.; ZHU, H.; RUAN, J.; QIAN, W.; FANG, X.; SHI, Z.; LI, Y.; LI, S.; SHAN, G.; KRISTIANSEN, K.; LI, S.; YANG, H.; WANG, J.; WANG, J.; WANG, J. (2010). De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2), 265–272.

LI, W. (2009). Analysis and comparison of very large metagenomes with fast clustering and functional annotation. *BMC Bioinformatics*, 10(1), 359.

LIAO, Y.; SMYTH, G. K.; SHI, W. (2014). featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics* (Oxford, England), 30(7), 923–930.

LIU, B.; LIU, C.-M.; LI, D.; LI, Y.; TING, H.-F.; YIU, S.-M.; … LAM, T.-W. (2016). BASE: a practical de novo assembler for large genomes using long NGS reads. *BMC Genomics*, 17 Suppl 5(S5), 499.

LOMSADZE, A.; TER-HOVHANNISYAN, V.; CHERNOFF, Y. O.; BORODOVSKY, M. (2005). Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Research*, 33(20), 6494–6506.

LU, H.; GIORDANO, F.; NING, Z. (2016). Oxford nanopore MinION sequencing and genome assembly. *Genomics, Proteomics & Bioinformatics*, 14(5), 265–279.

LUKASHIN, A. V.; BORODOVSKY, M. (1998). GeneMark.hmm: new solutions for gene finding. *Nucleic Acids Research*, 26(4), 1107–1115.

MARDIS, E. R. (2011). A decade's perspective on DNA sequencing technology. *Nature*, 470(7333), 198–203.

MARGULIES, M.; EGHOLM, M.; ALTMAN, W. E.; ATTIYA, S.; BADER, J. S.; BEMBEN, L. A.; BERKA, J.; BRAVERMAN, M. S.; CHEN, Y.; CHEN, Z.; DEWELL, S. B.; DU, L.; FIERRO, J. M.; GOMES, X. V.; GODWIN, B. C.; HE, W.; HELGESEN, S.; HO, C.

H.; IRZYK, G. P.; JANDO, S. C.; ALENQUER, M. L.; JARVIE, T. P.; JIRAGE, K. B.; KIM, J. B.; KNIGHT, J. R.; LANZA, J. R.; LEAMON, J. H.; LEFKOWITZ, S. M.; LEI, M.; LI, J.; LOHMAN, K. L.; LU, H.; MAKHIJANI, V. B.; MACDADE, K. E.; MCKENNA, M. P.; MYERS, E. W.; NICKERSON, E.; NOBILE, J. R.; PLANT, R.; PUC, B. P.; RONAN, M. T.; ROTH, G. T.; SARKIS, G. J.; SIMONS, J. F.; SIMPSON, J. W.; SRINIVASAN, M.; TARTARO, K. R.; TOMASZ, A.; VOGT, K. A.; VOLKMER, G. A.; WANG, S. H.; WANG, Y.; WEINER, M. P.; TU, P.; BEGLEY, R. F.; ROTHBERG, J. M. (2005). Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057), 376–380.

MATHÉ, C.; SAGOT, M.-F.; SCHIEX, T.; ROUZÉ, P. (2002). Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research*, 30(19), 4103–4117.

MAXAM, A. M.; GILBERT, W. (1977). A new method for sequencing DNA. *Proceedings of the National Academy of Sciences of the United States of America*, 74(2), 560–564.

MCCLINTOCK, B. T.; LANGROCK, R.; GIMENEZ, O.; CAM, E.; BORCHERS, D. L.; GLENNIE, R.; PATTERSON, T. A. (2020). Uncovering ecological state dynamics with hidden Markov models. *Ecology Letters*, 23(12), 1878–1903.

MEDVEDEV, P.; PHAM, S.; CHAISSON, M.; TESLER, G.; PEVZNER, P. (2011). Paired de bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 18(11), 1625–1634.

MEYER, F.; PAARMANN, D.; D'SOUZA, M.; OLSON, R.; GLASS, E. M.; KUBAL, M.; PACZIAN, T.; RODRIGUEZ, A.; STEVENS, R.; WILKE, A.; WILKENING, J.; EDWARDS, R. A. (2008). The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics*, 9(1), 386.

MEYER, FOLKER, BAGCHI, S.; CHATERJI, S.; GERLACH, W.; GRAMA, A.; HARRISON, T.; PACZIAN, T.; TRIMBLE, W. L.; WILKE, A. (2019). MG-RAST version 4—lessons learned from a decade of low-budget ultra-high-throughput metagenome analysis. *Briefings in Bioinformatics*, 20(4), 1151–1159.

MILLER, J. R.; KOREN, S.; SUTTON, G. (2010). Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6), 315–327.

MISTRY, J.; CHUGURANSKY, S.; WILLIAMS, L.; QURESHI, M.; SALAZAR, G. A.; SONNHAMMER, E. L. L.; TOSATTO, S. C. E.; PALADIN, L.; RAJ, S.; RICHARDSON, L. R.; FINN, R. D.; BATEMAN, A. (2021). Pfam: The protein families database in 2021. *Nucleic Acids Research*, 49(D1), D412–D419.

MOHAMED, M. A.; GADER, P. (2000). Generalized hidden Markov models. I. Theoretical frameworks. *IEEE transactions on fuzzy systems: a publication of the IEEE Neural Networks Council*, 8(1), 67–81.

MÖLDER, F.; JABLONSKI, K. P.; LETCHER, B.; HALL, M. B.; TOMKINS-TINCH, C. H.; SOCHAT, V.; FORSTER, J.; LEE, S.; TWARDZIOK, S. O.; KANITZ, A.; WILM, A.; HOLTGREWE, M.; RAHMANN, S.; NAHNSEN, S.; KÖSTER, J. (2021). Sustainable data analysis with Snakemake. *F1000Research*, 10, 33.

NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION. (26 de febrero de 2021). *Assembly*. https://www.ncbi.nlm.nih.gov/assembly/help/.

NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION. (s.f). *Blast topics.* https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp.

NATIONAL HUMAN GENOME RESEARCH INSTITUTE. (21 de junio de 2021). *Metagenómica*. https://www.genome.gov/es/genetics-glossary/Metagenomica.

NATIONAL HUMAN GENOME RESEARCH INSTITUTE. (s.f.) *Cóntigo* (contig). https://www.genome.gov/es/genetics-glossary/C%C3%B3ntigo.

NATIONAL HUMAN GENOME RESEARCH INSTITUTE. (s.f.) *Marco abierto de lectura.* https://www.genome.gov/es/genetics-glossary/Marco-abierto-de-lectura.

NF-CORE. (20 de julio de 2020). *Introduction to DSL2 in Nextflow - Evan Floden & Paolo Di Tommaso* [Archivo de video]. Youtube. https://www.youtube.com/watch?v=I-hunuzsh6A

NING, Z.; COX, A. J.; MULLIKIN, J. C. (2001). SSAHA: a fast search method for large DNA databases. *Genome Research*, 11(10), 1725–1729.

NOGUCHI, H.; TANIGUCHI, T.; ITOH, T. (2008). MetaGeneAnnotator: detecting species-specific patterns of ribosomal binding site for precise gene prediction in anonymous prokaryotic and phage genomes. DNA Research: An International Journal for Rapid *Publication of Reports on Genes and Genomes*, 15(6), 387–396.

NURK, S.; MELESHKO, D.; KOROBEYNIKOV, A.; PEVZNER, P. A. (2017). metaSPAdes: a new versatile metagenomic assembler. Genome research, 27(5), 824–834.

OGATA, H., GOTO, S.; SATO, K.; FUJIBUCHI, W.; BONO, H.; KANEHISA, M. (1999). KEGG: Kyoto Encyclopedia of genes and genomes. *Nucleic Acids Research*, 27(1), 29–34.

OVERBEEK, R.; OLSON, R.; PUSCH, G. D.; OLSEN, G. J.; DAVIS, J. J.; DISZ, T.; EDWARDS, R. A.; GERDES, S.; PARRELLO, B.; SHUKLA, M.; VONSTEIN, V.; WATTAM, A. R.; XIA, F.; STEVENS, R. (2014). The SEED and the Rapid Annotation of microbial genomes using Subsystems Technology (RAST). *Nucleic Acids Research*, 42(Database issue), 206-214.

PENG, Y.; LEUNG, H. C. M.; YIU, S. M.; CHIN, F. Y. L. (2012). IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics* (Oxford, England), 28(11), 1420–1428.

PENNISI, E. (2010). Genomics. Semiconductors inspire new sequencing technologies. *Science* (New York, N.Y.), 327(5970), 1190.

PHILLIPPY, A. M.; SCHATZ, M. C.; POP, M. (2008). Genome assembly forensics: finding the elusive mis-assembly. *Genome Biology*, 9(3), 55.

POTTER, S. C.; CLARKE, L.; CURWEN, V.; KEENAN, S.; MONGIN, E.; SEARLE, S. M. J.; STABENAU, A.; STOREY, R.; CLAMP, M. (2004). The Ensembl analysis pipeline. *Genome Research*, 14(5), 934–941.

PRIMMER, C. R.; PAPAKOSTAS, S.; LEDER, E. H.; DAVIS, M. J.; RAGAN, M. A. (2013). Annotated genes and nonannotated genomes: cross-species use of Gene Ontology in ecology and evolution research. *Molecular Ecology*, 22(12), 3216–3241.

PRJIBELSKI, A. D.; VASILINETC, I.; BANKEVICH, A.; GUREVICH, A.; KRIVOSHEEVA, T.; NURK, S.; PHAM, S.; KOROBEYNIKOV, A.; LAPIDUS, A.; PEVZNER, P. A. (2014). ExSPAnder: a universal repeat resolver for DNA fragment assembly. *Bioinformatics* (Oxford, England), 30(12), 293-301.

REHM, H. L.; BALE, S. J.; BAYRAK-TOYDEMIR, P.; BERG, J. S.; BROWN, K. K., DEIGNAN, J. L.; FRIEZ, M. J.; FUNKE, B. H.; HEDGE, M. R.; LYON, E.; WORKING GROUP OF THE AMERICAN COLLEGE OF MEDICAL GENETICS AND GENOMICS LABORATORY QUALITY ASSURANCE COMMITEE. (2013). ACMG clinical laboratory standards for next-generation sequencing. *Genetics in Medicine: Official Journal of the American College of Medical Genetics*, 15(9), 733-747.

RHO, M.; TANG, H.; YE, Y. (2010). FragGeneScan: predicting genes in short and error-prone reads. *Nucleic Acids Research*, 38(20), 191.

RHOADS, A.; AU, K. F. (2015). PacBio Sequencing and its applications. *Genomics, Proteomics & Bioinformatics*, 13(5), 278–289.

RICHARDSON, E. J.; WATSON, M. (2013). The automatic annotation of bacterial genomes. *Briefings in Bioinformatics*, 14(1), 1–12.

ROBINSON, M. D.; MCCARTHY, D. J.; SMYTH, G. K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* (Oxford, England), 26(1), 139–140.

RUMBLE, S. M.; LACROUTE, P.; DALCA, A. V.; FIUME, M.; SIDOW, A.; BRUDNO, M. (2009). SHRiMP: accurate mapping of short color-space reads. *PLoS Computational Biology*, 5(5), e1000386.

SAIKI, R.; GELFAND, D.; STOFFEL, S.; SCHARF, S. J.; HIGUCHI, R.; HORN, G. T.; MULLIS, K. B.; ERLICH, H. A. (1988). Primer-directed enzymatic amplification of DNA with a thermostable DNA polymerase. *Science* (New York, N.Y.), 239(4839), 487–491.

SANGER, F.; & COULSON, A. R. (1975). A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology*, 94(3), 441–448.

SANGER, F.; NICKLEN, S.; COULSON, A. R. (1977). DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, 74(12), 5463–5467.

SANGWAN, N.; XIA, F.; GILBERT, J. A. (2016). Recovering complete and draft population genomes from metagenome datasets. *Microbiome*, 4(1), 8.

SCHADT, E. E.; TURNER, S.; KASARSKIS, A. (2010). A window into third-generation sequencing. *Human Molecular Genetics*, 19(R2), 227-240.

SCHATZ, M. C. (2009). CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics* (Oxford, England), 25(11), 1363–1369.

SEEMANN, T. (2014). Prokka: rapid prokaryotic genome annotation. *Bioinformatics* (Oxford, England), 30(14), 2068–2069.

SELENGUT, J. D.; HAFT, D. H., DAVIDSEN, T.; GANAPATHY, A.; GWINN-GIGLIO, M.; NELSON, W. C.; RICHTER, A. R.; WHITE, O. (2007). TIGRFAMs and Genome Properties: tools for the assignment of molecular function and biological process in prokaryotic genomes. *Nucleic Acids Research*, 35(Database issue), 260-264.

SEQUERA LABS. (2020) *NEXTFLOW'S DOCUMENTATION! — Nextflow 21.04.1 documentation.* Barcelona: Centre for Genomic Regulation (CRG). https://www.nextflow.io/docs/latest/index.html.

SERVANT, F.; BRU, C.; CARRÈRE, S.; COURCELLE, E.; GOUZY, J.; PEYRUC, D.; KAHN, D. (2002). ProDom: automated clustering of homologous domains. *Briefings in Bioinformatics*, 3(3), 246–251.

SMITH, T. F.; WATERMAN, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1), 195–197.

SMITH, T.; HEGER, A.; SUDBERY, I. (2017). UMI-tools: modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. *Genome Research*, 27(3), 491–499.

SONNHAMMER, E. L.; EDDY, S. R.; DURBIN, R. (1997). Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins*, 28(3), 405–420.

STEIN, L. (2001). Genome annotation: from sequence to biology. *Nature Reviews*. Genetics, 2(7), 493–503.

TAMAMES, J.; PUENTE-SÁNCHEZ, F. (2018). SqueezeMeta, A highly portable, fully automatic metagenomic analysis pipeline. *Frontiers in Microbiology*, 9, 3349.

TATUSOV, R. L.; FEDOROVA, N. D.; JACKSON, J. D.; JACOBS, A. R.; KIRYUTIN, B.; KOONIN, E. V.; KRYLOV, D. M.; MAZUMDER, R.; MEKHEDOV, S. L.; NIKOLSKAYA, A. N.; RAO, B. S.; SMIRNOV, S.; SVERDLOV, A. V.; VASUDEVAN, S.; WOLF, Y. I.; YIN, J. J.; NATALE, D. A. (2003). The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4, 41.

THE SAM/BAM FORMAT SPECIFICATION WORKING GROUP. (2021). *Sequence alignment/map format specification.* Cambridge: Sanger Institute. http://samtools.github.io/hts-specs/SAMv1.pdf.

THOMAS, T.; GILBERT, J.; MEYER, F. (2012). Metagenomics - a guide from sampling to data analysis. *Microbial Informatics and Experimentation*, 2(1), 3.

TRAPNELL, C.; ROBERTS, A.; GOFF, L.; PERTEA, G.; KIM, D.; KELLEY, D. R.; PIMENTEL, H.; SALZBERG, S. L.; RINN, J. L.; PACHTER, L. (2012). Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature Protocols*, 7(3), 562–578.

TRAPNELL, C.; WILLIAMS, B. A.; PERTEA, G.; MORTAZAVI, A.; KWAN, G.; VAN BAREN, M. J.; SALZBERG, S. L.; WOLD, B. J.; PACHTER, L. (2010). Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5), 511–515.

TREANGEN, T. J.; SOMMER, D. D.; ANGLY, F. E.; KOREN, S.; POP, M. (2011). Next generation sequence assembly with AMOS. *Current Protocols in Bioinformatics*, 11(1), 11.8.

UNIVERSITAT POLITÈCNICA DE VALENCIA. (4 de octubre de 2017). *Modelos de Markov ocultos || UPV* [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=lnOkyvWcAtQ.

VASILINETC, I.; PRJIBELSKI, A. D.; GUREVICH, A.; KOROBEYNIKOV, A.; PEVZNER, P. A. (2015). Assembling short reads from jumping libraries with large insert sizes. *Bioinformatics* (Oxford, England), 31(20), 3262–3268.

VELOSO, C. (18 de abril de 2018). MODELO DE ARQUITECTURA PIPELINE. https://www.electrontools.com/Home/WP/modelo-de-arquitectura-pipeline/.

VOELKERDING, K. V.; DAMES, S. A.; DURTSCHI, J. D. (2009). Next-generation sequencing: from basic research to diagnostics. *Clinical Chemistry*, 55(4), 641–658.

WANG, Z.; GERSTEIN, M.; SNYDER, M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews. Genetics*, 10(1), 57–63.

WEISSTEIN, E. W. *Acyclic Digraph.* MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/AcyclicDigraph.html.

WILKE, A.; BISCHOF, J.; GERLACH, W.; GLASS, E.; HARRISON, T.; KEEGAN, K. P.; PACZIAN, T.; TRIMBLE, W. L.; BAGCHI, S.; GRAMA, A.; CHATERJI, S.; MEYER, F. (2016). The MG-RAST metagenomics database and portal in 2015. *Nucleic Acids Research*, 44(D1), 590-594.

WILKE, A.; BISCHOF, J.; HARRISON, T.; BRETTIN, T.; D'SOUZA, M.; GERLACH, W.; MATTHEWS, H.; PACZIAN, T.; WILKENING, J.; GLASS, E. M.; DESAI, N.; MEYER, F. (2015). A RESTful API for accessing microbial community data for MG-RAST. *PLoS Computational Biology*, 11(1), e1004008.

ZERBINO, D. R.; BIRNEY, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5), 821–829.

ZHAO, S.; FUNG-LEUNG, W. P.; BITTNER, A.; NGO, K.; LIU, X. (2014). Comparison of RNA-Seq and microarray in transcriptome profiling of activated T cells. *PloS One*, 9(1), e78644.

ZHU, W.; LOMSADZE, A.; BORODOVSKY, M. (2010). Ab initio gene identification in metagenomic sequences. *Nucleic Acids Research*, 38(12), 132.

7. ANEXO I

FORMATOS DE ARCHIVO

FORMATO FASTA

Un archivo FASTA se reconoce por las extensiones ". fasta, .fna, .ffn, .faa, .frn".
Este formato se emplea para representar aquellas secuencias provenientes, por ejemplo, de procesos de secuenciación, en forma de ácidos nucleicos o péptidos. Visualmente, se pueden encontrar dos partes separadas: la primera línea o "*defline*" aparece señalada por el símbolo (">") y consta de la descripción de la secuencia en una única línea, siendo la primera palabra adyacente al símbolo (">") el identificador de la secuencia. El resto de las líneas representan la secuencia mediante códigos de una letra, códigos estándar de ácidos nucleicos y aminoácidos de la IUB/IUPAC, la Figura 9 ejemplifica el formato FASTA. No obstante, hay excepciones como la posibilidad de utilizar guiones para denotar un *gap – con longitud indeterminada*– o el uso de minúsculas.

```
>P01013 GENE X PROTEIN (OVALBUMIN-RELATED)
QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNSFNVATLPAE
KMKILELPFASGDLSMLVLLPDEVSDLERIEKTINFEKLTEWTNPNTMEKRRVKVYLPQMKIEEKYNLTS
VLMALGMTDLFIPSANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESEQFRADHP
FLFLIKHNPTNTIVYFGRYWSP
```

Figura 9. Ejemplo visual de un archivo formato FASTA (NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION [NCBI], s.f.).

En el caso de los archivos denominados multi-FASTA, lo que aparece son una serie de secuencias en formato FASTA.

FORMATO FASTQ

FASTQ es un formato de archivo que integra lecturas provenientes de secuenciación, y la puntuación de la calidad, asociada a dichas lecturas, base por base.
Se representa con la extensión ".fastq", tiene dos variantes con respecto al estándar original, SANGER, que se denominan variantes Solexa e Illumnia. No obstante, la versión de SANGER de FASTQ es la que ha obtenido la hegemonía, siendo apoyada por diferentes herramientas bioinformáticas como son SSAHA2 (Ning *et al*., 2001), MAQ, Velvet, BWA (Li y Durbin, 2009) and BowTie que se emplean en procesos de ensamblaje y mapeo.
En un archivo FASTQ aparecen 4 líneas distintas por lectura. La primera línea comienza con una arroba ('@'), a la que le sigue un identificador de registro (ID). Adicionalmente pueden describirse otras características como la longitud de la secuencia, comentarios o identificaciones adicionales; siendo de esta forma un campo sin límite de caracteres y de libre escritura. La segunda, corresponde con la secuencia "problema", no existe limitación de caracteres y se suele emplear el uso de mayúsculas. Aunque, en algunos casos aparecen minúsculas, e incluso ambas. La tercera línea corresponde con el final de la línea de secuencia y el comienzo de la línea de calidad. Generalmente, esta línea contiene únicamente el símbolo "+". En los primeros archivos FASTQ la tercera línea contenía, además del símbolo antes mencionado, una repetición del ID plasmado en la primera línea que, por motivos de tamaño de archivo, ha quedado eliminada con el paso del tiempo. La última de las cuatro líneas presenta la calidad de secuenciación de cada uno de los nucleótidos que componen la secuencia. Por lo tanto, esta línea y la segunda deben tener la misma extensión. Los símbolos que marcan la calidad pertenecen al subconjunto de caracteres imprimibles ASCII. Entre ellos aparece la arroba, lo que implica que, en el caso de que una línea comience por "@", la herramienta que analice el archivo debe comprobar la longitud de dicha línea para no que no haya una confusión con la línea de cabecera (Cock *et al*., 2010).
Toda la explicación anterior se reduce visualmente en la Figura 10.

```
@ML-P2-14:9:000H003HG:1:11102:17290:1073 1:N:0:TCCTGAGC+GCGATCTA
TTTGGTAACAGCATGAATTATTCTAGCCACTAAAACTCTATGAACATCTTGTGAAGGTTTCAGATAGAGCCTGAAGTACACAGAGAACAATTCTTAAAAAA
+
AAAAAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE<AEEEEEEE
```

Figura 10. Ejemplo visual de un archivo formato FASTQ (ILLUMINA, 2020a).

FORMATO YALM

YALM es un acrónimo recursivo de *YAML Ain't Markup Language* que en castellano se traduce como "YAML no es un lenguaje de marcado" (https://yaml.org/), y los archivos de este tipo se nombran mediante la extensión ".ylm". El objetivo del formato reside en facilitar la legibilidad de los archivos, al mismo tiempo que permitir su empleabilidad por la amplia mayoría de lenguajes de más usados en el mundo de la informática como Python, Java, JavaScript, Perl o Haskell. En resumen, YALM es capaz de mostrar cualquier tipo de dato mediante una combinación de datos escalares, listas y funciones resumen (función hash) (YAML AIN'T MARKUP LANGUAGE VERSION 1.2, 2009). Los archivos YAML se emplean con predominancia en archivos de configuración.

Como ejemplo se exponen los archivos YAML pertenecientes al pipeline presentado en la tesis Figura 11 y 12.

```yaml
name: nf-mtg_diamond-0.9.22
channels:
  - default
  - conda-forge
  - bioconda
dependencies:
  - python=2.7
  - bioconda::diamond=0.9.22
```

Figura 11. Archivo "diamond.ylm" que contiene las instrucciones para crear el ambiente conda que alberga el paquete de *software* DIAMOND (0.9.22).

```yaml
name: prokka1.14.6
channels:
  - conda-forge
  - bioconda
  - defaults
dependencies:
  - prokka=1.14.6
```

Figura 12. Archivo "prokka.ylm" que contiene las instrucciones para crear el ambiente conda que alberga el paquete de *software* PROKKA (1.14.6).

FORMATO SAM/BAM

SAM o *Sequence Alignment/Map format* es un tipo de archivo que se encuentra delimitado por tabulaciones (*TAB-delimited text*) y se denota por la extensión ".sam". El esquema de este formato se divide en las líneas de cabecera que son opcionales y están precedidas por el símbolo "@", y las líneas de alineamiento. Estas últimas están compuestas por once campos obligatorios que se detallan en la Tabla 5 y se representan en la Figura 13.

Además, existen otros campos opcionales que pueden aparecen en un archivo SAM que pueden aparecer en cualquier orden.

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,254}`	Query template NAME
2	FLAG	Int	$[0, 2^{16} - 1]$	bitwise FLAG
3	RNAME	String	`\*|[:rname:^*=][:rname:]*`	Reference sequence NAME[11]
4	POS	Int	$[0, 2^{31} - 1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0, 2^{8} - 1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[:rname:^*=][:rname:]*`	Reference name of the mate/next read
8	PNEXT	Int	$[0, 2^{31} - 1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31} + 1, 2^{31} - 1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

Tabla 5. Descripción de los campos obligatorios que aparecen en un archivo con formato SAM.
[11]Los nombres de las secuencias de referencia pueden contener cualquier carácter ASCII imprimible, a excepción de algunos caracteres de puntuación, y no pueden empezar por "*" o "=" (THE SAM/BAM FORMAT SPECIFICATION WORKING GROUP, 2021).

```
@HD VN:1.6 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

Figura 13. Ejemplo visual de un archivo formato SAM (THE SAM/BAM FORMAT SPECIFICATION WORKING GROUP, 2021).

Por otro lado, el formato BAM se considera la versión binaria comprimida del formato SAM. Como ventaja, tiene la propiedad de que está diseñado para ser fácilmente comprimible, pero es más difícil de procesar y menos legible que los archivos SAM. Los archivos BAM también poseen una cabecera con información como el nombre de la muestra o la longitud, y una parte de alineamientos que contiene seis campos entre los que se encuentran el número de lecturas de cada muestra (RG), la calidad de alineamiento single-end y paired-end (SM/AS) o el nombre de la etiqueta del amplicón (XN) (ILLUMINA, s.f. a).

FORMATO GTF

Los archivos con extensión ".gtf", en inglés *General Transfer Format* , son el output típico de anotación que presenta la siguiente estructura (Figura 14) de nueve campos o columnas separadas por tabulador:

```
1 transcribed_unprocessed_pseudogene  gene        11869 14409 . + . gene_id "ENSG00000223972"; gene_name "DDX11L1"; gene_source "havana"; gene_biotype "transcribed_unprocessed_pse
udogene";
1 processed_transcript                transcript  11869 14409 . + . gene_id "ENSG00000223972"; transcript_id "ENST00000456328"; gene_name "DDX11L1"; gene_sourc e "havana"; gene_bi
otype "transcribed_unprocessed_pseudogene"; transcript_name "DDX11L1-002"; transcript_source "havana";
```

Figura 14. Ejemplo visual de un archivo formato GTF (ENSEMBL, s.f.).

La primera columna corresponde al cromosoma donde se encuentra el gen en cuestión. La segunda muestra la fuente de la que proviene la anotación (en este caso Ensembl); la tercera caracteriza la secuencia anotada, pudiendo ser etiquetada como: "start_codon", "CDS" o "stop_codon". Las dos columnas adyacentes marcan el inicio y el final de la secuencia problema, respectivamente. La siguiente columna, se denomina de score y se representa con un punto que "indica la posible presencia de señal del point floating value" (CRG.EU, sf).
La séptima columna marca la hebra de DNA que se transcribe para codificar dicha secuencia. Es decir, las opciones son "-" o "+". La penúltima define el marco de lectura (ORF) por el que se comienza a transcribir – 1, 2 o 3 – y la última, proporciona información sobre los identificadores de los genes y sus características (separadas por ";"), puede incluir su "E-value".

FORMATO GFF

El formato GFF, en inglés *gene-finding format* o *generic feature format*, es un archivo con extensión ".gff" creado con el objetivo de identificar y describir genes, sus características y secuencias proteicas.

La estructura de este tipo de archivos es muy similar a la de los GTF: tiene, del mismo modo, 9 campos separados por tabulador, que coinciden exactamente hasta la columna 7. Se diferencian solamente en el último campo (columna 9) que contiene los atributos de la secuencia, como se puede apreciar en la Figura 15.

Figura 15. Ejemplo visual de un archivo formato GFF (ENSEMBL, s.f.).

8. ANEXO II

CONCEPTOS

SECUENCIACIÓN *SINGLE-END*

La secuenciación *single-end* se refiere al proceso en el que se secuencia desde un único extremo del fragmento "problema". Este tipo de secuenciación es la más sencilla.

SECUENCIACIÓN CON LECTURAS PAREADAS

Secuenciación empleando la estrategia de leer ambos extremos de la lectura al mismo tiempo, siendo esta técnica más compleja que la *single-end* y genera bibliotecas de alta calidad. La mayoría de las técnicas de secuenciación de nueva generación (NGS) obtienen lecturas pareadas, existen dos estrategias que conforman este tipo de secuenciación: mate pairs y paired-end (ILLUMINA, 2020b).

ESTRATEGIA *MATE PAIRS*

La estrategia *mate pair* consiste en fragmentar las lecturas de DNA en fragmentos largos (>600 bp, hasta 5 kb). Estos se circularizan gracias a la biotinilización de sus extremos, y se re-fragmentan en fragmentos de entre 200 y 600 pb. Se purifican aquellos que tienen biotina – los extremos- por afinidad. Posteriormente se produce la ligación de los adaptadores de secuenciación y se secuencia, como se detalla en la Figura 16.

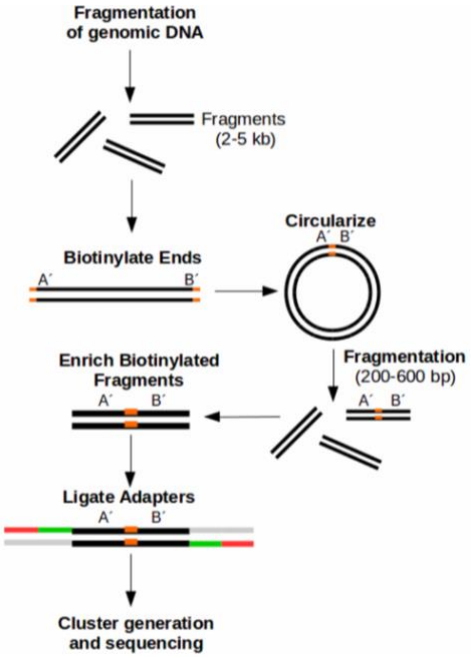

Figura 16. Esquema de la estrategia mate pair (ECSEQ BIOINFORMATICS, 2017).

Una ventaja de esta estrategia es que es capaz de cubrir rangos de mayor tamaño (ILLUMINA, s.f. b).

ESTRATEGIA *PAIRED END*

La estrategia *paired end* es sencilla, como se refleja en la Figura 17, y consiste en fragmentar en segmentos de un tamaño pequeño (<300pb) las lecturas a secuenciar; y posteriormente, se lee desde amos extremos del fragmento. El resultado es la posibilidad de cubrir tamaños de inserto más pequeños.

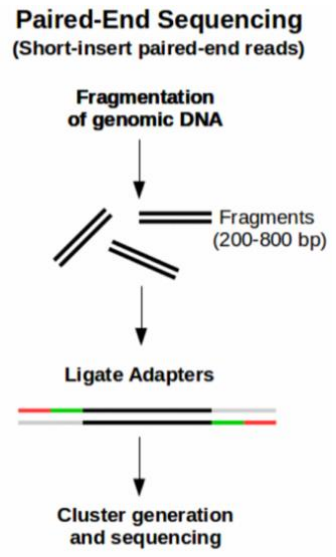

Figura 17. Esquema de la estrategia *paired end* (ECSEQ BIOINFORMATICS, 2017).

***SINGLE CELL SEQUENCING* (SCS)/ *SINGLE MOLECULE SEQUENCING* (SMS)**
La secuenciación de célula única o de células individuales en un tipo de secuenciación que abarca una variedad de tecnologías, y se emplea para estudiar células en solitario. A consecuencia, estas técnicas tienen una mayor resolución y se enmarcan en los métodos denominados HTS (*High Throughput Sequencing*) o secuenciación de alto rendimiento. La secuenciación por síntesis (SBS) y la secuenciación a tiempo real (SMRT) son tipos de SCS (Eberwine *et al*., 2014).

***SEQUENCING BY SYNTHESIS* (SBS)**
La secuenciación por síntesis es un tipo de secuenciación masiva (NGS) que fue lanzado al mercado por Illumina. La peculiaridad de la SBS es el uso de la amplificación clonal *in vitro* gracias a una PCR en puente. El enfoque de la secuenciación por sínstesis se emplea para efectuar la SCS (ILLUMINA, s.f. c).

***SINGLE MOLECULE REAL TIME SECUENCING* (SMRT)**
En castellano secuenciación a tiempo real de una única molécula de DNA, este tipo de secuenciación de tercera generación (TSG) fue desarrollada por Pacific Biosciences. Su funcionamiento consiste en el empleo de un chip – que presenta una estructura ZWM (guía de onda "modo cero") – en el que se distribuyen moléculas de DNA polimerasa de forma individual y sDNA (DNA de cadena sencilla). En otras palabras, el chip está dividido en pozos microscópicos en los que se aloja una DNA polimerasa activa y un sDNA molde. Dicha tecnología la secuenciación se acelera y se obtienen fragmentos bastante largos (>7kb) (Eid *et al*., 2009).

***POLYMERASE CHAIN REACTION* (PCR)**
La PCR o reacción en cadena de la polimerasa es una técnica de laboratorio, perteneciente al campo de la biología molecular, cuyo objetivo es la amplificación de secuencias de DNA con fines de investigación y diagnósticos. El procedimiento requiere de diferentes reactivos como son primers (cebadores), una solución tampón (buffer), DNA polimerasa (enzima que produce la amplificación) y sus cofactores, desoxirribonucleótidos-trifosfato (dNTP) y, por supuesto, la muestra de DNA molde. El resultado son amplificar el número de copias de la hebra molde para otros estudios (Saiki *et al*., 1998).

***OPEN READING FRAME* (ORF)**
Un ORF, en castellano marco de lectura abierto, es la secuencia que se enmarca entre un codón de inicio (en eucariotas predomina el codón "AUG") y un codón de terminación. Es decir, es un

fragmento que al pasar por el proceso de traducción no posee codones de terminación. Los ORFs suelen ser exones que forman parte de un gen. En concreto existen seis sentidos en los que aparecn marcos abiertos de lectura (+1, +2, +3, -1,-2,-3), lo que se debe a la naturaleza de la lectura del código genético que ocurre en codones de 3 bp (NHGRI, s.f. a).

cDNA
El cDNA o DNA complementario es una molécula sintetizada – artificialmente – por medio de la enzima transcriptasa reversa y que corresponde a una copia proveniente del RNA mensajero. Como consecuencia, estos transcritos no tienen intrones.

***EXPRESSED SEQUENCE TAG* (EST)**
En castellano marcador de secuencia expresada, "es una pequeña sub-secuencia de una secuencia nucleotídica transcripta (codificante de una proteína o no). Se pueden usar para identificar genes que se transcriben y en el descubrimiento de genes, y para determinación de secuencias" (Adams *et al.*, 1991).
Estas "etiquetas" se producen mediante la secuenciación de un ARNm clonado, por lo que se consideran de baja calidad. GenBank (base de datos pública) alberga más de 50 millones de ESTs diferentes.

ADAPTADORES
En el contexto de las nuevas técnicas secuenciación masiva (NGS) los adaptadores se definen como secuencias consenso que reconocen los *primers* (cebadores) que se emplean para la amplificación y la secuenciación en sí. En concreto son fragmentos/secuencias de doble cadena de DNA y son específicos para cada plataforma de secuenciación (Rehm *et al*., 2013).

RNA-seq
RNA-seq también conocida como *Whole Transcriptome Shotgun Sequencing* o, en castellano, Secuenciación del Transcriptoma para Clonación al Azar, es una técnica de secuenciación que permite secuenciar el producto cDNA de RNA mensajeros de una muestra.
Actualmente, esta técnica se usa para identificar los transcriptos de una célula, para experimentos de expresión diferencial o para conocer los modelos e isoformas de las proteínas (Chu y Corey, 2012).

CONTIG
Conjunto de lecturas de DNA que se solapan, gracias a las cuáles se pueden conseguir secuencias más largas. Los contigs se obtienen tras el proceso de ensamblaje y representan una región consenso del genoma (NHGRI, s.f. b).

SCAFFOLD
El término tiene su análogo en castellano, denominándose en esta lengua andamio. No obstante, es comúnmente empleada la palabra *scaffold* en el campo de las ciencias ómicas.
En concreto los *scaffold* son un conjunto de *contigs* ordenados, pero en este caso no tienen porqué solaparse entre sí, siendo normal la aparición de huecos entre unos y otros (NCBI, 2021).

BINNING
El *data binning* consiste en un proceso de pre-procesamiento de los datos a analizar, con el objetivo de facilitar el análisis posterior. En metagenómica se refiere a la agrupación de lecturas o contigs en bins para su posterior asignación a un genoma individual. Dependiendo el tipo de enfoque, se da el binning antes o después del alineamiento: si se produce antes el objetivo del proceso son las lecturas, gracias al agrupamiento se reducen las necesidades computacionales, pero el resultado del alineamiento se vuelve menos fiable por la pequeña longitud de los bins.
El binning se puede dar con diferentes estrategias: agrupación por identidad (similitud), agrupación por composición, o ambas (Sangwan *et al*., 2016).

FM-INDEX

El índice FM es un índice que "aprovecha la relación entre la transformada Burrows-Wheeler y la estructura de datos de matriz de sufijos" (Ferragina y Manzini, 2005). El objetivo de esta herramienta es conseguir pasar de un texto comprimido a un índice que represente todo el contenido del archivo gracias a la Transformación Burrows-Wheeler (BWT).

K-MER

Un k-mer es una subcadena de longitud k que se encuentra contenida dentro de una secuencia biológica.

GRAFOS DE BRUJIN

"Un grafo De Bruijin se describe como un grafo dirigido que representa los solapamientos de longitud (n-1) entre todas las palabras de longitud n con un alfabeto dado" (Bruijn, 1946), Figura 18.

En el campo de la bioinformática se emplean estos grafos en herramientas de ensamblado. Las lecturas para ensamblar se fragmentan en "palabras" de longitud n que se solapan mediante estos grafos.

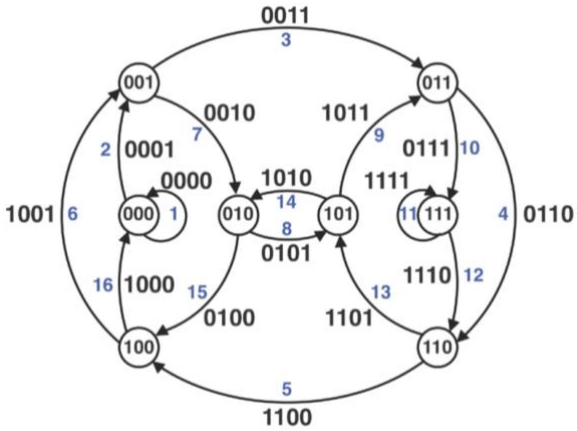

Figura 18. Ejemplo visual del grafo de Bruijn. El grafo de Bruijn B para n = 4 y un alfabeto de dos caracteres compuesto por los dígitos 0 y 1. Este grafo tiene un ciclo euleriano (un camino euleriano se define como el camino que pasa por cada arista una vez) porque cada nodo tiene un grado igual a 2. Siguiendo las aristas numeradas en azul en orden del 1 al 16 se traza un ciclo euleriano 0000, 0001, 0011, 0110, 1100, 1001, 0010, 0101, 1011, 0111, 1111, 1110, 1101, 1010, 0100, 1000. Al registrar el primer carácter (en negrita) de cada etiqueta de borde deletrea la supercadena cíclica 0000110010111101 (Compeau, P. *et al*., 2011).

GRAFOS EMPAREJADOS DE BRUJIN (PDBG)

"Los grafos de Bruijn emparejados son una generalización de los grafos de Brujin mediante la incorporación de la información que proporciona la secuenciación *mate pair*" (Medvedev *et al*., 2011). La Figura 19 aclara las diferencias entre los grafos de Bruijn y los PDGB.

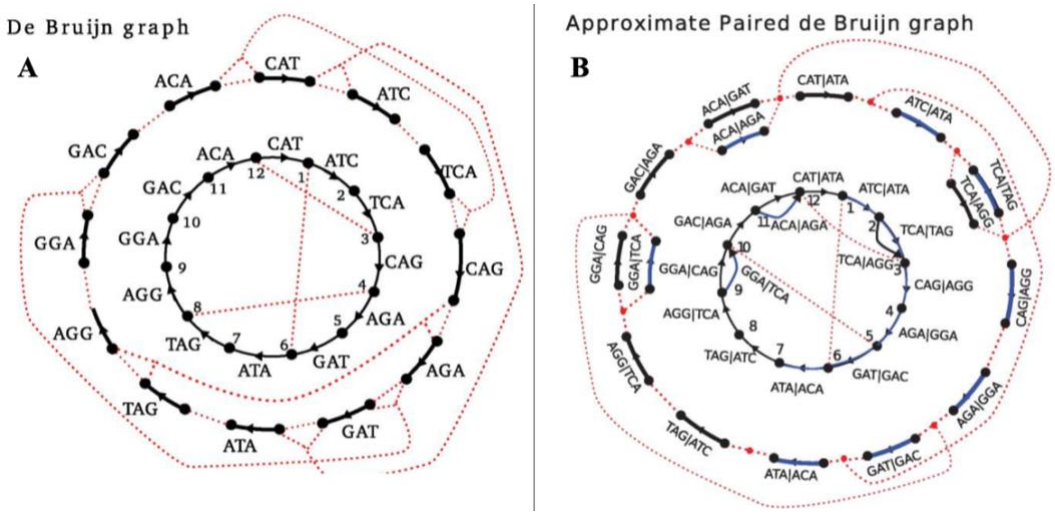

Figura 19. Comparación entre un grafo de Bruijn (A) y un grafo emparejado de Bruijn (B). En la imagen A, el círculo exterior muestra un borde negro separado para cada k-mer (k = k-mers de 3 nucleótidos de longitud). Las líneas rojas punteadas indican los vértices que se solapan. El círculo interior muestra el resultado de aplicar algunas de las colas. En la imagen B se muestra un posible espectro de cobertura en el círculo exterior, con aristas negras para los elementos con distancia entre *mate pairs* de 6 y aristas azules para la distancia de 5 (Medvedev *et al*., 2011).

GRAFO ACÍCLICO DIRIGIDO (DAG)

Un grafo acíclico dirigido es un grafo dirigido pero que no contiene ciclos directos; es decir, no existe una ruta directa que empiece y termine en un mismo vértice del grafo como queda representado en la Figura 20 (Weisstein, s.f.).

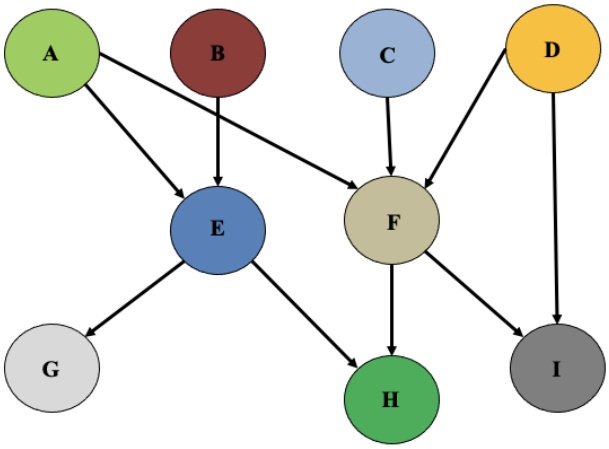

Figura 20. Representación de un DAG, los nodos se han nombrado aleatoriamente.