

# Máster en Bioinformática

Secuenciación Genómica y Análisis De Variantes Para Medicina Personalizada y De Precisión

Curso académico 2024-25  
Edición Abril



**Universidad**  
Internacional  
de Valencia

Dra. Laura Gutiérrez Macías  
[laura.gutierrez.m@professor.universidadviu.com](mailto:laura.gutierrez.m@professor.universidadviu.com)

# Capítulo 5. Análisis de genomas bacterianos

## 5.1.

- **Genoma procariota.**
  - 5.1.1.- Características generales de los genomas procariotas
  - 5.1.2.- Genomas multipartitos
  - 5.1.3.- Estructura detallada del genoma procariota.
  - 5.1.4.- Número de genes y función general

## 5.2.

- **Epidemiología genómica. Retos y oportunidades en el análisis de genomas bacterianos.**

## 5.3.

- **Tipificación de genomas bacterianos**
  - 5.3.1.- Análisis basado en polimorfismos de único nucleótido (SNP)
  - 5.3.2.- Análisis gen por gen
  - 5.3.3.- Identidad nucleotídica promedio (Average Nucleotide Identity, ANI)

## 5.4.

- **Reconstrucción del genoma: ensamblaje *de novo***

- 5.4.1.- Genotecas para el ensamblado de novo
- 5.4.2.- Conceptos generales de ensamblaje
- 5.4.3.- Tipos de algoritmos de ensamblaje.
- 5.4.4.- Evaluación previa al ensamblaje. Análisis de k-meros.
- 5.4.5.- Ensamblaje de secuencias cortas con SPAdes
- 5.4.6.- Evaluación de la calidad del ensamblaje: continuidad y contenido

## 5.5.

- **Reconstrucción del genoma: anotación del genoma ensamblado.**

- 5.5.1.- Anotación utilizando Prokka
- 5.5.2.- Visualización

- 5.6.
  - Detección y anotación de regiones de interés: genes de resistencia a antibióticos, virulencia, biocidas y metales pesados.
- 5.7.
  - Reconstrucción del genoma: Elementos Genéticos Móviles (EGMs). Plásmidos.
    - 5.6.1.- Herramientas de ensamblaje y extracción de plásmidos
    - 5.6.2.- Herramientas de identificación mediante marcadores genéticos.
    - 5.6.3.- Herramientas basadas en los grafos de ensamblaje de novo.
- 5.8. EXTRA!
  - Calidad, filtrado y ensamblaje de lecturas cortas de 3a generación (ONT).

# Environment (datos Illumina\*)

Instalación Mamba:	conda install mamba -n base -c conda-forge				
Entorno*	Programa	Comando descarga	Versión	Utilidad	Anotaciones (extras a instalar, referencias...)
04MBIF_bacteriano	<code>conda create -n 04MBIF_bacteriano python=3.8 mamba conda 04MBIF_bacteriano</code>				
	FastQC	conda install -c bioconda fastqc	v0.12.1	Análisis primario de calidad	
	MultiQC	conda install -c bioconda multiqc	v1.19	Integrar archivos de calidad, mapeo, etc de diferentes análisis	
	FastP	conda install -c bioconda fastp	v0.23.4	Limpieza de adaptadores y trimming	
	Spades	conda install -c bioconda spades	v3.15.5	Ensamblaje de novo	
	Quast	conda install -c bioconda quast	v5.2.0	Evaluación de ensamblajes	Para funcionalidad completa es necesario bajar varios accesorios: quast-download-gridss quast-download-silva quast-download-busco [error de descarga, hay que hacer descarga manual]
	Prokka	conda install -c conda-forge -c bioconda -c defaults prokka	v1.14.6	Anotación de genomas	Antes de utilizarlo <i>puede</i> que sea necesario configurar las variables export PERL5LIB=\$CONDA_PREFIX/lib/perl5/site_perl/5.22.0/ conda env config vars set PERL5LIB=\$CONDA_PREFIX/lib/perl5/site_perl/5.22.0/ -n 04MBIF_bacteriano conda deactivate conda activate 04MBIF_bacteriano
	Artemis	conda install -c bioconda artemis	v18.2.0	Visualización de anotaciones	
	Bandage	conda install -c bioconda bandage_ng	v2022.09	Visualización del grafo de ensamblaje	Si hay error de instalación en conda, bajar desde web: <a href="https://rrwick.github.io/Bandage/">https://rrwick.github.io/Bandage/</a>
	AMRFinderPlus	conda install -c bioconda ncbi-amrfinderplus	v3.11.26	Detección genes de resistencia a antibióticos, metales pesados y genes de virulencia	Antes de utilizarlo realizar: amrfinder_update
<code>conda env export --file=04MBIF_bacteriano.yml</code>					

\*Usaremos otro environment para los datos de ONT

# Introducción

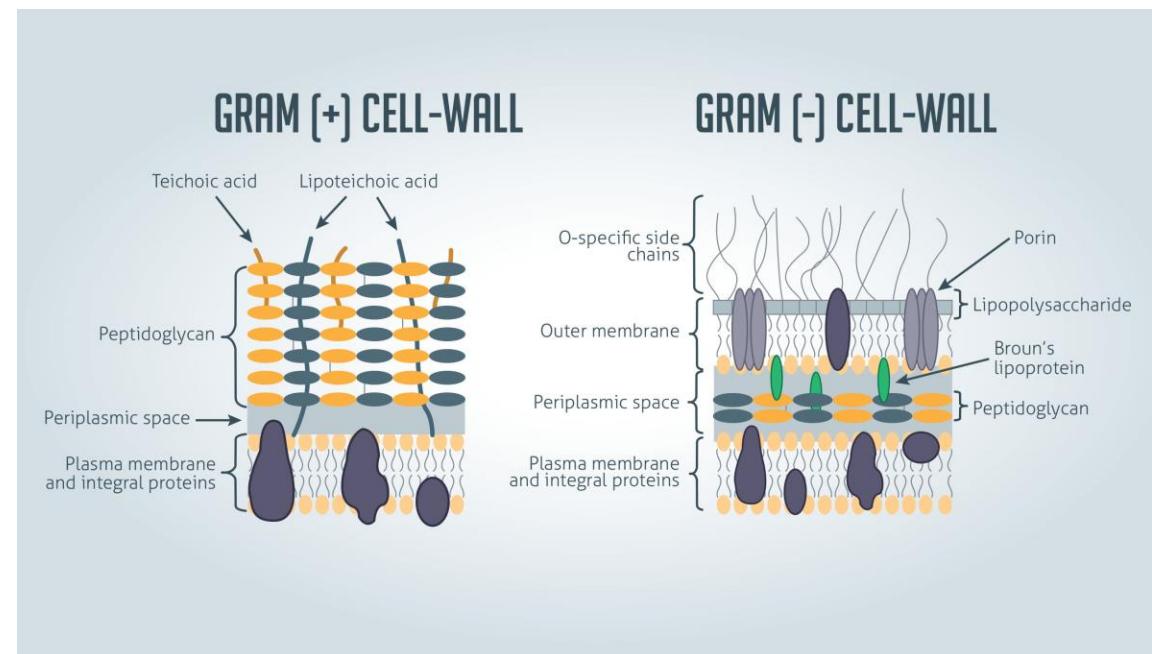
A lo largo de este capítulo se ahondará en el análisis global de genomas bacterianos, especialmente enfocado desde el punto de vista de **la epidemiología genómica** y la monitorización de bacterias de interés bien por sus características de resistencia, virulencia o patogenia. La metodología que se describirá será aplicable a cualquier genoma bacteriano que queramos analizar.

# 5.1.

# Genoma procaríota

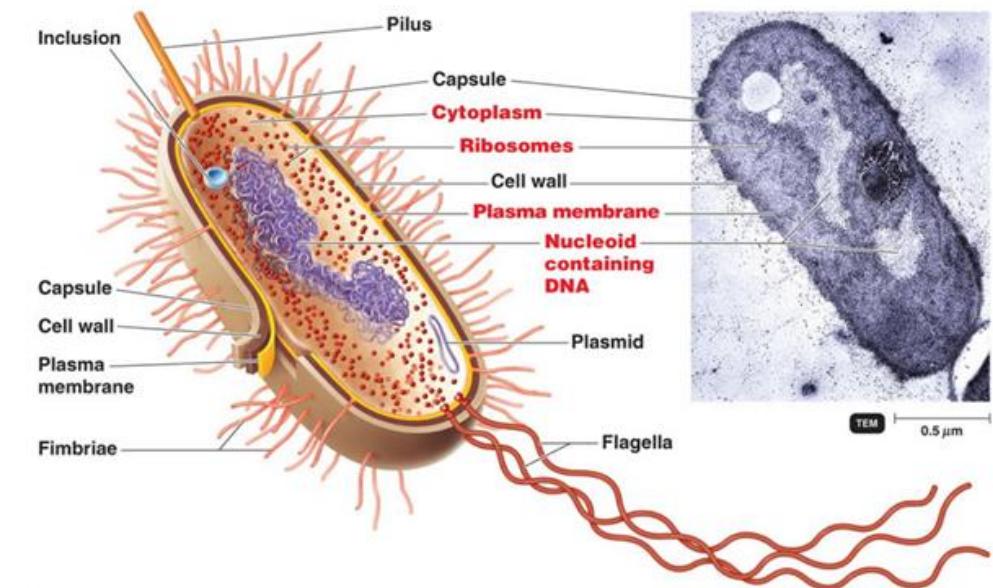
## 5.1.1. Características generales de los genomas procariotas

De manera general, los procariotas son organismos cuyas células **carecen de compartimentos internos**. Diferenciamos dos tipos de procariotas que varían en sus características genéticas y bioquímicas. Las bacterias incluyen a la mayor parte de los procariotas como son las bacterias **gramnegativas** (por ejemplo, *Escherichia coli*), **grampositivas** (*Bacillus subtilis*), **cianobacterias** (*Anabaena*), y otras más. Por otra parte, las Archaea son un grupo mucho menos estudiado que ha sido encontrado principalmente en ambientes extremos.



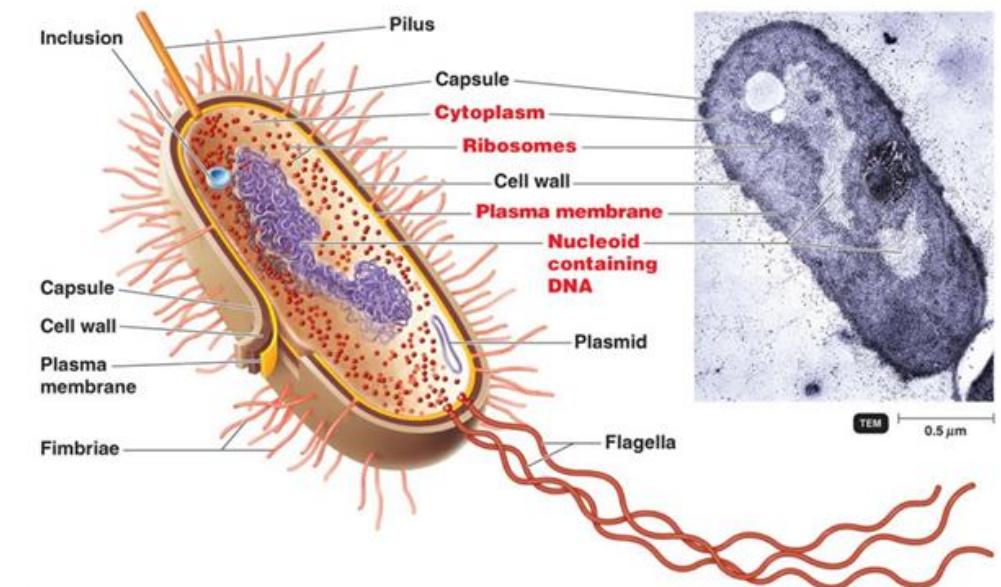
## 5.1.1. Características generales de los genomas procariotas

De manera general, el genoma procariota típico está contenido en una única molécula de ADN circular localizada en el nucleoide, una región físicamente no separada del citosol. No existe una membrana formando un núcleo separado, a diferencia de lo que ocurre en eucariotas. Esta diferencia principal es muy habitual en un gran número de bacterias, sobre todo aquellas más estudiadas, como *E. coli*. Sin embargo, el conocimiento creciente nos hace cuestionar algunos de estos conceptos. Al igual que ocurre en eucariotas, existe una serie de proteínas nucleares que participan en el empaquetamiento del genoma de una manera organizada con forma superenrollada (supercoiled).



## 5.1.1. Características generales de los genomas procariotas

A medida que se conocen más genomas procariotas sabemos que la amplia mayoría de bacterias cumplen esta estructura de cromosoma. Sin embargo, cada vez conocemos más estructuras de **cromosomas lineares**, como son los genomas de *Borrelia burgdorferi* (causante de la enfermedad de Lyme), *Streptomyces coelicolor* y *Agrobacterium tumefaciens*. Estas moléculas de ADN lineales tienen extremos libres, con una terminación similar a lo que son los telómeros en los eucariotas.



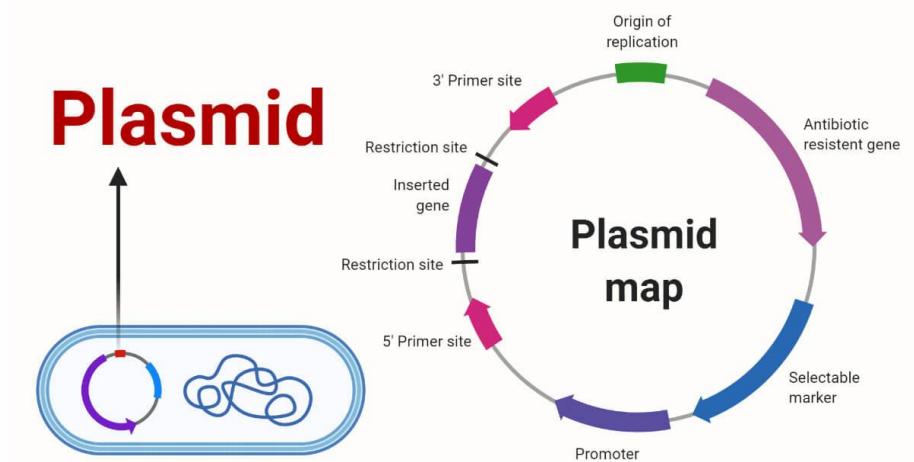
## 5.1.2. Genomas multipartitos

Podemos destacar que existen procariotas que poseen genomas **multipartitos**, es decir, genomas divididos en dos o más moléculas de ADN.

En estos genomas lo complicado radica en distinguir entre lo que corresponde al cromosoma, a lo que corresponde a elementos genéticos independientes como son los plásmidos.

Brevemente, un **plásmido** es un elemento genético pequeño, habitualmente circular, que coexiste con el cromosoma principal en la célula bacteriana y que codifica para factores como genes de resistencia a antibióticos, metales pesados o factores de virulencia. Además, estos elementos son transferibles de manera horizontal entre bacterias.

Hablaremos en profundidad sobre ellos en un apartado posterior, ya que van a ser una parte fundamental de nuestro estudio bioinformático.



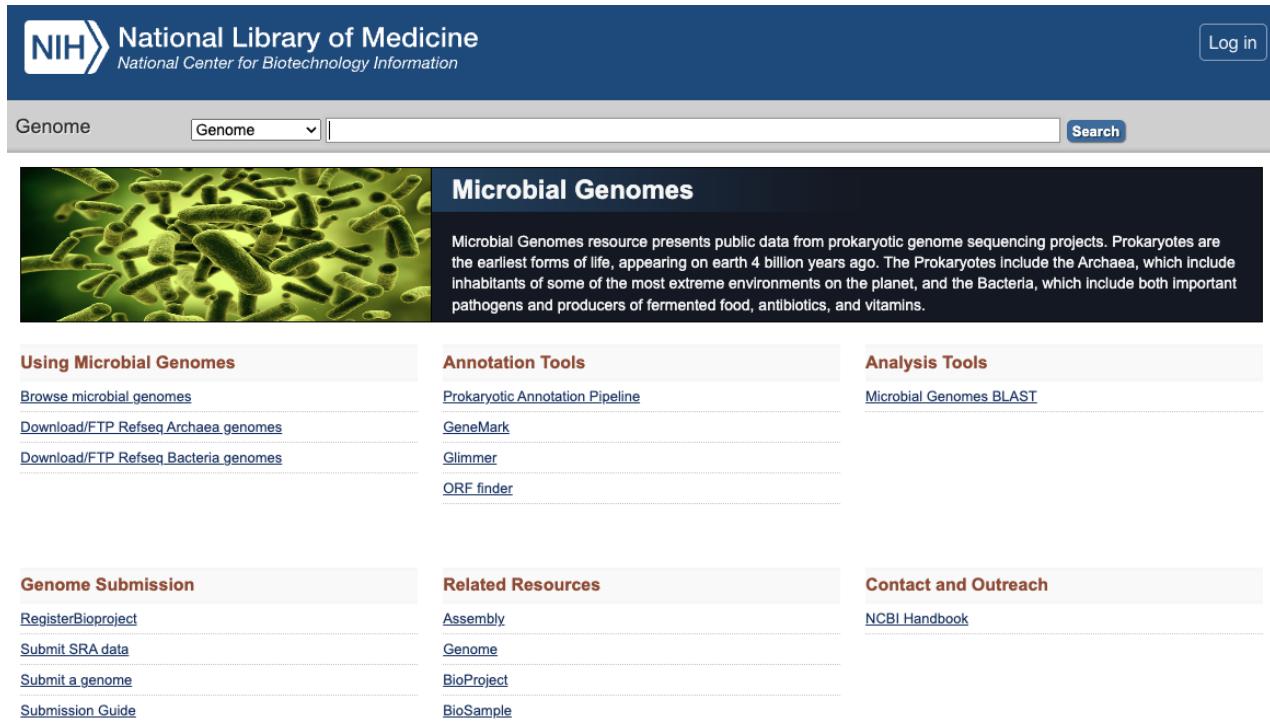
## 5.1.2. Genomas multipartitos

Especie bacteriana	Molécula de ADN	Tamaño (Mb)	Número de genes
<i>Escherichia coli</i> K12	Cromosoma	4,639	4405
<i>Vibrio cholerae</i> El Tor N16961	Cromosoma	2,961	2770
	Megaplásmido	1,073	1115
<i>Deinococcus radiodurans</i> R1	Cromosoma 1	2,649	1633
	Cromosoma 2	0,412	369
	Megaplásmido	0,177	145
	Plásmido	0,046	40
<i>Borrelia burgdorferi</i> B31	Cromosoma lineal	0,911	853
	Plásmido circular cp9	0,009	12
	Plásmido circular cp26	0,026	29
	Plásmido circular cp32	0,032	No conocido
	Plásmido lineal lp17	0,017	25
	Plásmido lineal lp25	0,024	32
	Plásmido lineal lp28-1	0,027	32
	Plásmido lineal lp28-2	0,030	34
	Plásmido lineal lp28-3	0,029	41
	Plásmido lineal lp28-4	0,027	43
	Plásmido lineal lp36	0,037	54
	Plásmido lineal lp38	0,039	52
	Plásmido lineal lp54	0,054	76
	Plásmido lineal lp56	0,056	No conocido

Para explorar los genomas procariotas junto con sus características, se puede encontrar la información actualizada en los siguientes enlaces:

<https://www.ncbi.nlm.nih.gov/genome/microbes/>

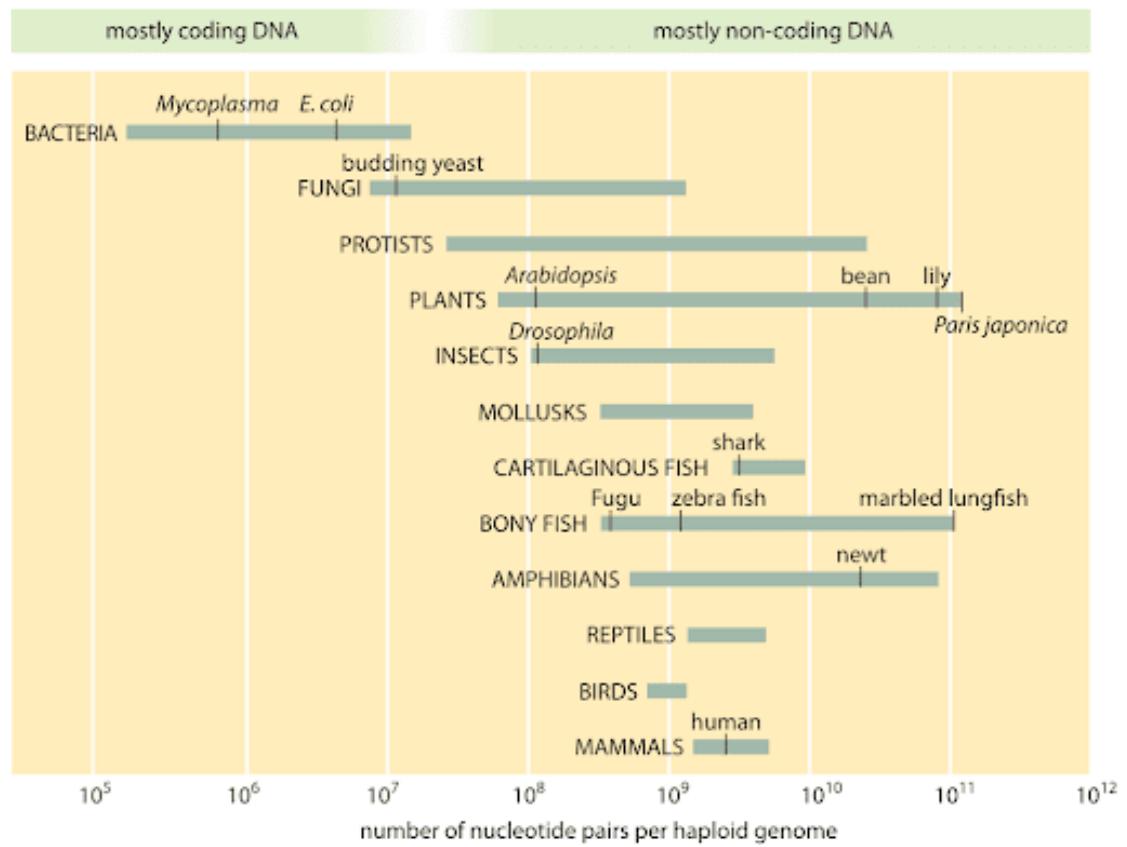
<https://www.ncbi.nlm.nih.gov/genome/browse#!/overview/>



The screenshot shows the homepage of the National Library of Medicine's Microbial Genomes resource. The header features the NIH logo and the text "National Library of Medicine" and "National Center for Biotechnology Information". A "Log in" button is in the top right. Below the header is a search bar with dropdown menus for "Genome" and "Species". To the left is a green image of various prokaryotic cells. The main content area is titled "Microbial Genomes" and includes a brief description of the resource, mentioning prokaryotes, archaea, and bacteria. Below this are three columns of links: "Using Microbial Genomes" (Browse microbial genomes, Download/FTP Refseq Archaea genomes, Download/FTP Refseq Bacteria genomes), "Annotation Tools" (Prokaryotic Annotation Pipeline, GeneMark, Glimmer, ORF finder), and "Analysis Tools" (Microbial Genomes BLAST). At the bottom are sections for "Genome Submission" (RegisterBioproject, Submit SRA data, Submit a genome, Submission Guide) and "Related Resources" (Assembly, Genome, BioProject, BioSample), along with a "Contact and Outreach" section (NCBI Handbook).

## 5.1.3. Estructura detallada del genoma procariota

Si nos fijamos en la organización de los genes en los genomas procariotas, encontramos que la característica principal es que los genes **no contienen intrones**; así como que existe muy poco espacio entre los distintos genes, haciendo un **genoma muy compacto**. Se estima que solo el 11 % del genoma del organismo modelo *E. coli* K12 es no codificante. Esto permite que la replicación de los genomas procariotas sea muy rápida.



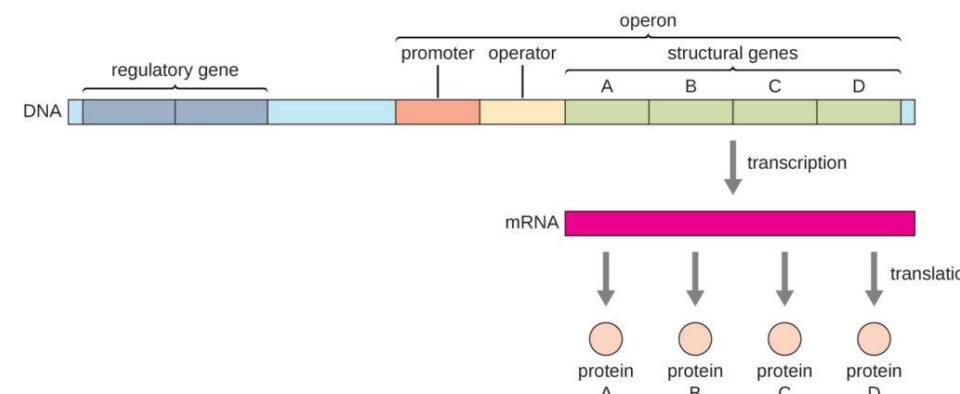
<https://book.bionumbers.org/how-big-are-genomes/>

organism	genome size (base pairs)	protein coding genes	number of chromosomes
<b>model organisms</b>			
<i>model bacteria E. coli</i>	4.6 Mbp	4,300	1
<i>budding yeast S. cerevisiae</i>	12 Mbp	6,600	16
<i>fission yeast S. pombe</i>	13 Mbp	4,800	3
<i>amoeba D. discoideum</i>	34 Mbp	13,000	6
<i>nematode C. elegans</i>	100 Mbp	20,000	12 (2n)
<i>fruit fly D. melanogaster</i>	140 Mbp	14,000	8 (2n)
<i>model plant A. thaliana</i>	140 Mbp	27,000	10 (2n)
<i>moss P. patens</i>	510 Mbp	28,000	27
<i>mouse M. musculus</i>	2.8 Gbp	20,000	40 (2n)
<i>human H. sapiens</i>	3.2 Gbp	21,000	46 (2n)
<b>viruses</b>			
hepatitis D virus (smallest known animal RNA virus)	1.7 Kb	1	ssRNA
HIV-1	9.7 kbp	9	2 ssRNA (2n)
<i>influenza A</i>	14 kbp	11	8 ssRNA
bacteriophage $\lambda$	49 kbp	66	1 dsDNA
<i>Pandoravirus salinus</i> (largest known viral genome)	2.8 Mbp	2500	1 dsDNA
<b>organelles</b>			
mitochondria - <i>H. sapiens</i>	16.8 kbp	13 (+22 tRNA +2 rRNA)	1
mitochondria - <i>S. cerevisiae</i>	86 kbp	8	1
chloroplast - <i>A. thaliana</i>	150 kbp	100	1
<b>bacteria</b>			
<i>C. ruddii</i> (smallest genome of an endosymbiont bacteria)	160 kbp	182	1
<i>M. genitalium</i> (smallest genome of a free living bacteria)	580 kbp	470	1
<i>H. pylori</i>	1.7 Mbp	1,600	1
<i>Cyanobacteria S. elongatus</i>	2.7 Mbp	3,000	1
methicillin-resistant <i>S. aureus</i> (MRSA)	2.9 Mbp	2,700	1
<i>B. subtilis</i>	4.3 Mbp	4,100	1
<i>S. cellulosum</i> (largest known bacterial genome)	13 Mbp	9,400	1
<b>archaea</b>			
<i>Nanoarchaeum equitans</i> (smallest parasitic archaeal genome)	490 kbp	550	1
<i>Thermoplasma acidophilum</i> (flourishes in pH<1)	1.6 Mbp	1,500	1
<i>Methanocaldococcus (Methanococcus) jannaschii</i> (from ocean bottom hydrothermal vents; pressure >200 atm)	1.7 Mbp	1,700	1
<i>Pyrococcus furiosus</i> (optimal temp 100°C)	1.9 Mbp	2,000	1
<b>eukaryotes - multicellular</b>			
pufferfish <i>Fugu rubripes</i> (smallest known vertebrate genome)	400 Mbp	19,000	22
poplar <i>P. trichocarpa</i> (first tree genome sequenced)	500 Mbp	46,000	19
corn <i>Z. mays</i>	2.3 Gbp	33,000	20 (2n)
dog <i>C. familiaris</i>	2.4 Gbp	19,000	40
chimpanzee <i>P. troglodytes</i>	3.3 Gbp	19,000	48 (2n)
wheat <i>T. aestivum</i> (hexaploid)	16.8 Gbp	95,000	42 (2n=6x)
marbled lungfish <i>P. aethiopicus</i> (largest known animal genome)	130 Gbp	unknown	34 (2n)
herb plant <i>Paris japonica</i> (largest known genome)	150 Gbp	unknown	40 (2n)

## 5.1.3. Estructura detallada del genoma procariota

Si centráramos nuestro interés en el genoma de esta bacteria y comparásemos un fragmento de su genoma frente a un fragmento del genoma humano, observaríamos:

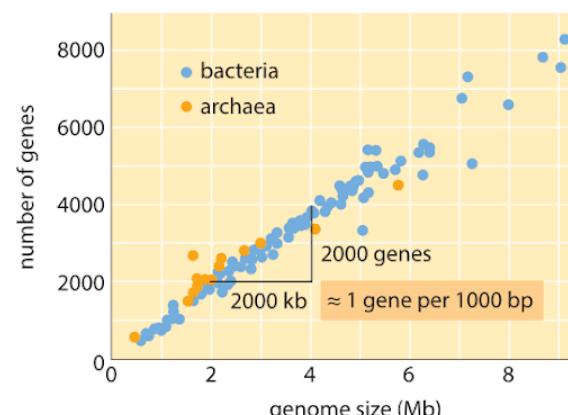
- Entre los genes de una bacteria existe **menos espacio** que entre los genes eucariotas, y que incluso algunos de ellos son solapantes, formando organizaciones llamadas **operones**. Un operón es un grupo de genes que están involucrados en una única ruta bioquímica y que se regulan conjuntamente.
- Los **genes bacterianos son más cortos** en longitud que los eucariotas, incluso sin tener en cuenta los intrones de estos últimos.
- **En general, no existen intrones en los genes bacterianos**, aunque existen algunas excepciones en el grupo de las Archaea.
- Aunque en el genoma procariota **existen regiones repetidas**, como son las secuencias de inserción, **estas no tienen la extensión, la frecuencia, ni el alto número de copias de las que se presentan en los genomas eucariotas**. Se considera que la mayor parte de los genomas procariotas poseen muy pocos elementos repetidos, como puede ser el genoma de *Campylobacter jejuni* NCTC11168, donde en sus 1,64 Mb de genoma no tiene ningún elemento repetido. Sin embargo, *Neisseria meningitidis* Z2491 tiene más de 3700 copias de 15 tipos diferentes de secuencias repetidas, lo que agrupa un 11 % de su genoma (2,18 Mb).



## 5.1.4. Número de genes y función general

Los genomas procariotas tienen en general un **tamaño de genoma** menor que un organismo eucariota, aunque el rango entre el genoma procariota más pequeño (*Nanoarchaeum equitans*, 491 kb) y el más grande secuenciado (*Bradyrhizobium japonicum*, 9,1 Mb) es amplio.

En cuanto al **número de genes**, la densidad de genes media es de aproximadamente 950 genes por cada megabase (Mb) de genoma.



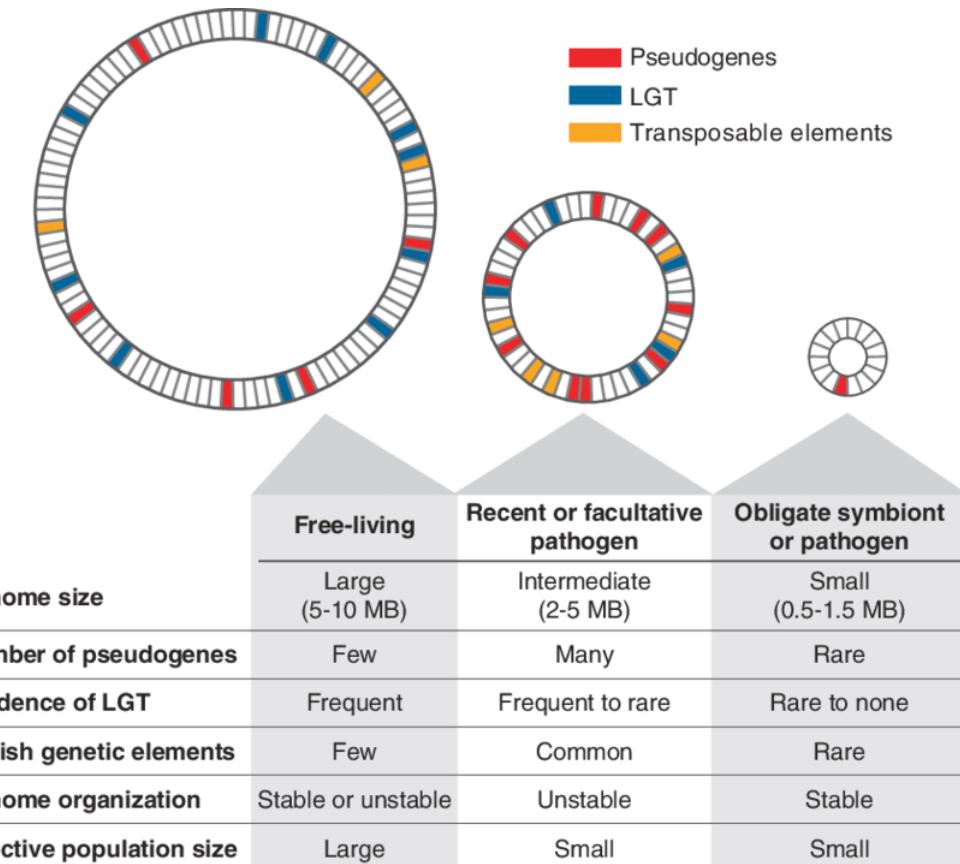
Organisms	Source	Genome size (Mb)	Total genes <sup>1</sup>
<b>Phage/viruses, organelle</b>			
1 mitochondrion	Human mitochondrion	$16.6 \times 10^3$	40
2 SV40	Simian virus 40	$5.2 \times 10^3$	8
3 TMV	Tobacco Mosaic Virus (RNA)	$6.4 \times 10^3$	4
4 HIV	AIDS virus (RNA)	$9.3 \times 10^3$	10
5 adenovirus 2	Human adenovirus	$35.9 \times 10^3$	11
6 $\lambda$ phage	lysogenic phage	$48.5 \times 10^3$	50
7 T4 phage	DNA virus E. coli	$169 \times 10^3$	300
<b>Bacteria</b>			
8 Mycoplasma genitalium		$580 \times 10^3$	470
9 Rickettsia prowazekii		$1.11 \times 10^6$	834
10 Streptococcus pyogenes		$1.85 \times 10^6$	1752
11 Bacillus subtilis		$4.21 \times 10^6$	4100
12 Escherichia coli		$4.64 \times 10^6$	4288
13 Mycobacterium tuberculosis		$4.41 \times 10^6$	3924
14 Helicobacter pylori		$1.668 \times 10^6$	1590 <sup>1</sup>
		$1.644 \times 10^6$	1495 <sup>1</sup>
<b>Eucaryotes</b>			
15 Saccharomyces cerevisiae	budding yeast	$12 \times 10^6$	5885
16 Caenorhabditis elegans	worm, Nematoda	$97 \times 10^6$	19100
17 Arabidopsis thaliana	plant, Angiospermae	$120 \times 10^6$	25500
18 Drosophila melanogaster	fruit fly, Diptera	$165 \times 10^6$	13000
19 Mus musculus	mouse	$2.5 \times 10^9$	30000
20 Homo sapiens	humans	$3.2 \times 10^9$	31000

<sup>1</sup> Two strains isolated from two geographically separated regions of USA (32).

## 5.1.4. Número de genes y función general

Los **genomas más grandes** tienden a pertenecer a especies de vida libre, encontrados en ambientes como suelos o aguas, donde necesitan un arsenal de funciones para hacer frente a situaciones físicas y biológicas muy cambiantes (p. ej., *E. coli* ambiental).

Por otro lado, los **genomas más pequeños** se piensa que son aquellos de especies que son parásitos obligados, tales como *Mycoplasma genitalium*. Esta capacidad limitada de codificar funciones en estos genomas pequeños queda suplida por la obtención de nutrientes de sus hospedadores o incluso de sus compañeros microbianos en ese ambiente. De acuerdo con este tipo de análisis se ha establecido que el **rango de genes indispensables para la vida procariota está en torno a 250-350**, dependiendo de la especie.



Especie	Tamaño de genoma (Mb)	Número de genes (aprox)
<b>Bacteria</b>		
<i>Mycoplasma genitalium</i>	0,58	500
<i>Streptococcus pneumoniae</i>	2,16	2300
<i>Vibrio cholerae</i> El Tor N16961	4,03	4000
<i>Mycobacterium tuberculosis</i> H37Rv	4,41	4000
<i>Escherichia coli</i> K12	4,64	4400
<i>Yersinia pestis</i> CO92	4,65	4100
<i>Pseudomonas aeruginosa</i> PAO1	6,26	5700
<b>Archaea</b>		
<i>Methanococcus jannaschii</i>	1,66	1750
<i>Archaeoglobus fulgidus</i>	2,18	2500

Categoría	Número de genes	
	<i>E. coli</i> K12	<i>M. genitalium</i>
Total de genes codificantes	4288	470
Biosíntesis de aminoácidos	131	1
Biosíntesis de cofactores	103	5
Biosíntesis de nucleótidos	58	19
Proteínas de envuelta celular	237	17
Metabolismo energético	243	31
Metabolismo intermediario	188	6
Metabolismo de lípidos	48	6
Replicación, recombinación y reparación de ADN	115	32
Plegamiento de proteínas	9	7
Proteínas regulatorias	178	7
Transcripción	55	12
Traducción	182	101
Captación de moléculas del ambiente	427	34

## 5.2.

Epidemiología genómica. Retos y oportunidades en el análisis de  
genomas bacterianos

Las técnicas de secuenciación de genoma completo es actualmente la herramienta más potente en análisis de genomas bacterianos con fines epidemiológicos.

La **epidemiología genómica** es la disciplina que ha evolucionado desde la epidemiología molecular, y que nos permite utilizar el genoma completo de un organismo como marcador epidemiológico de su evolución y trazado.

De esta manera, **el análisis de los genomas completos nos permite trazar la transmisión de los microorganismos con la mayor resolución posible**. Aunque ha sido la pandemia de SARS-CoV-2 la que ha mostrado al mundo el potencial de estas técnicas, la epidemiología genómica ya ha sido ampliamente utilizada para la monitorización de patógenos de interés clínico, como son bacterias resistentes a los antibióticos (Comas, 2017; Deng et al., 2016).

En el siguiente vídeo, elaborado por el Centro para la Prevención y el Control de Enfermedades (CDC) de EE. UU. se explican conceptos básicos de epidemiología genómica.

<https://www.youtube.com/watch?v=njS0Tw4uQUo>

La secuenciación del genoma completo de los patógenos nos ha ayudado a resolver **brotes epidémicos**:

- Alto poder discriminatorio para diferenciar aislados estrechamente relacionados
- Rastrear tendencias epidemiológicas mediante la monitorización de resistencia a antibióticos: eje central en la epidemiología genómica bacteriana.

El desarrollo de la epidemiología genómica ha venido de la mano del desarrollo en las tecnologías de secuenciación masiva.

El primer genoma bacteriano secuenciado completamente fue el de *Haemophilus influenzae* Rd en 1995, mediante tecnología Sanger (Fleischmann et al., 1995). Desde ese momento y hasta la actualidad, existe en la base de datos RefSeq de National Center for Biotechnology Information (NCBI) un elevado número de genomas procariotas

En este enlace tenéis acceso a la base de datos RefSeq de NCBI.

<https://www.ncbi.nlm.nih.gov/genome/browse#!/overview/>

Esta explosión en el número de genomas ha venido dada por el rápido avance en el desarrollo de tecnologías de secuenciación como Illumina, SOLiD-Ion Torrent o Roche 454.

Estas tecnologías, **actualmente siendo el mayor exponente la tecnología Illumina**, tienen una **limitación fundamental**: las **lecturas de secuenciación cortas no son suficientes para cubrir los elementos repetidos, como operones ARN ribosómicos o transposones multicopia**. Consecuentemente, **los genomas que están ensamblados a partir de este tipo de lecturas están fragmentados** y consisten en decenas o centenas de fragmentos de ADN, llamados **contigs**.

Aquellas tecnologías, como **Oxford Nanopore o PacBio**, que **producen lecturas de secuenciación más largas pueden resolver estas repeticiones y permiten, en combinación con lecturas cortas de mayor resolución, ensamblajes híbridos menos fragmentados**. Sin embargo, como punto negativo a estas tecnologías está aún su alta tasa de error\*, que nos dificulta la identificación y tipificación genómica con detalle, como veremos en apartados posteriores.

\*Desde septiembre 2022 se encuentra disponible la química Q20+ para Nanopore (chips R10.4.1 + duplex reads), donde las tasas de error se han disminuido. Más información: <https://nanoporetech.com/q20plus-chemistry>



5.3.

Tipificación de genomas bacterianos

## Subtipado de bacterias con fines de trazado de brotes

Antes de la explosión de las NGS se usaban técnicas moleculares como, la determinación de perfiles de campos pulsados (pulse field gel electrophoresis, PFGE) o secuenciación de locus genéticos (multi locus sequence typing, MLST).

En los **últimos años se ha impuesto el análisis de genoma completo sobre estos métodos** que analizan tan solo una parte del genoma, **siendo ya una rutina** en países como Reino Unido, Dinamarca, Francia, EE. UU. y Canadá (Álvarez-Molina et al., 2021; Jagadeesan et al., 2019).

Determinar similitud genética entre patógenos se pueden seguir dos aproximaciones:

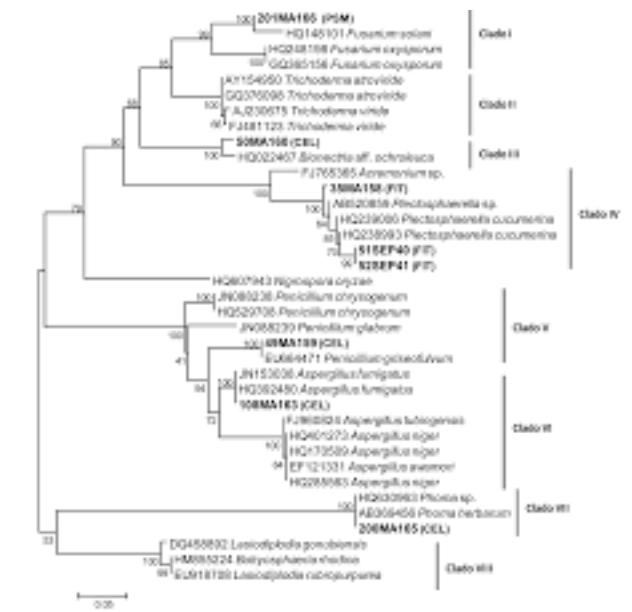
- Basado en polimorfismos de un único nucleótido (single nucleotide polymorphism, SNP)
- Análisis gen por gen.
- Adicionalmente: **análisis de identidad nucleotídica promedio** (average nucleotide identity, ANI), es un método de tipado e identificación bacteriano.

## 5.3.1. Análisis basado en polimorfismos de un único nucleótido (SNP)

Esta técnica se basa en el mapeo o alineamiento de las lecturas secuenciadas sobre un genoma de referencia bien conocido (**resecuenciación**). Las diferencias nucleotídicas entre cada par de genomas se analizan como variantes de secuencia, incluyendo cambios nucleotídicos, inserciones y delecciones. Estos cambios se utilizan para cuantificar la relación genética y por tanto para inferir una relación filogenética.

Aunque este tipo de análisis es muy potente, su debilidad y su punto clave es la selección del genoma de referencia adecuado, que debe estar completamente secuenciado y estar genéticamente relacionado con los genomas de estudio, para no infraestimar la similitud genética entre ellos.

Una de las mayores debilidades de esta técnica es la baja reproducibilidad entre laboratorios diferentes, ya que no solo debe usarse el mismo genoma de referencia y el mismo protocolo bioinformático, sino también los mismos parámetros de mapeo y llamada de variantes. Además, la interpretación de un árbol filogenético para evaluar un brote puede ser complejo y la metodología requiere pericia en el manejo de datos bioinformáticos.

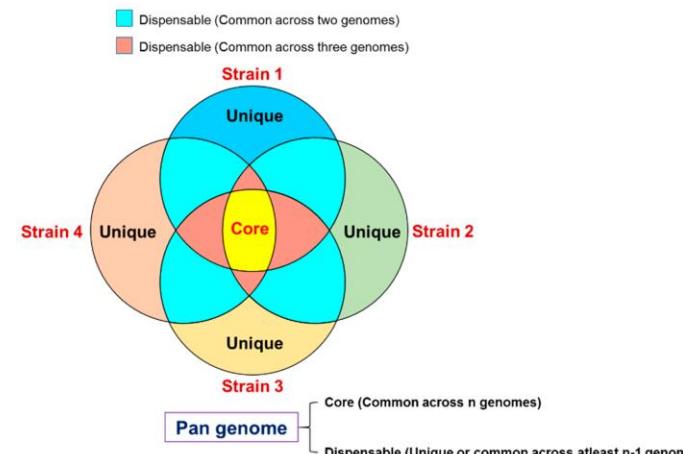


## 5.3.1. Análisis basado en polimorfismos de un único nucleótido (SNP)

Una buena guía de **interpretación de análisis filogenéticos** se encuentra en la referencia [Pightling et al. \(2018\)](#); adicionalmente, si queréis ver un ejemplo de cómo el genoma de referencia determina el resultado del análisis de un brote epidémico, podéis consultar el artículo de Francés-Cuesta et al. (2021) y Valiente-Mullor et al.

A diferencia del caso que vimos en el apartado de genomas humanos, donde los protocolos de análisis de SNP están bien definidos, en genomas bacterianos aún no hay un consenso. En este punto, los mapeadores más ampliamente utilizados son **Bowtie2** (Langmead y Salzberg, 2012) o **BWA** (Li y Durbin, 2010), en combinación con programas de llamada de variantes como SAMtools/bcftools (Li et al., 2009) o Freebayes (Garrison y Marth, 2012). Existen algunos protocolos integrados desarrollados como **Snippy** (Torsten Seemann, s. f.), uno de los más populares.

Otro de los métodos que se basan en la identificación de **SNP** conlleva generar alineamientos del **genoma core o del pangenoma**. En este caso primero hay que identificar los genes ortólogos o bloques de secuencias, alinearlos e identificar las localizaciones variables. En este punto podemos analizar regiones altamente variables que son sospechosas de recombinación. Este proceso puede llevarse a cabo con **Snippy** (Torsten Seemann, s. f.), **Parsnp/Harvesttools** (Treangen et al., 2014) o con **Roary** (Page et al., 2015).



## 5.3.2. Análisis gen por gen

Por otra parte, el análisis gen por gen (gene-by-gene) se basa en la aproximación tradicional de **tipado de genes multilocus (Multi Locus Sequence Typing, MLST)**. En esta técnica **se selecciona un grupo pequeño de genes** (tradicionalmente siete genes), que son **considerados housekeeping**, es decir, que cumplen funciones vitales.

Se amplifican mediante reacción en cadena de la polimerasa (PCR) y se secuencian **mediante técnica Sanger**. El esquema inicial de 7 loci, definido para varias especies bacterianas individualmente, se ha extendido para un esquema ribosomal (**rMLST, 53 genes**), un esquema core-genome (**cgMLST**), de genoma completo (**wgMLST**) y de genoma accesorio (**agMLST**). Estos incluyen más regiones variables esenciales o accesorias para cada especie.

El punto fuerte de estos esquemas de tipado es que **no dependen de la selección de un genoma de referencia cercano, sino que los alelos definidos para cada gen son fácilmente accesibles en bases de datos públicas**. En el caso de la utilización de datos de genoma completo, las lecturas se pueden ensamblar previamente (como veremos en los apartados posteriores), para posteriormente localizar estos genes y tipificarlos mediante un alineamiento frente a la base de datos de referencia (BLAST). **Estas bases de datos permiten la estandarización y comparación entre diferentes laboratorios, además implementados en soluciones comerciales de uso sencillo.**

*En el siguiente enlace se encuentra la base de datos pública de MLST:*

<https://pubmlst.org/>

<https://pubmlst.org/species-id>

### 5.3.3. Identidad nucleotídica promedio (average nucleotide identity, ANI)

Una de las cuestiones fundamentales en microbiología es conocer si existe un continuo de diversidad genética entre las distintas especies o, si en caso contrario, prevalecen unos límites claros entre especies.

En este punto, medir la similitud del genoma completo en forma de **identidad nucleotídica promedio (ANI)** ayuda a abordar esta cuestión y facilita el análisis taxonómico de alta resolución de miles de genomas de diversos linajes filogenéticos.

Esta medida, **dada como porcentaje de similitud**, nos **permite comparar genomas completos dos a dos** y **generar matrices de distancias y árboles filogenéticos**. El límite de especie se ha determinado en diversos estudios (Goris et al., 2007; Jain et al., 2018; Kim et al., 2014) **en un valor de ANI > 95 % la definición de especie** (Figueroa et al., 2014).

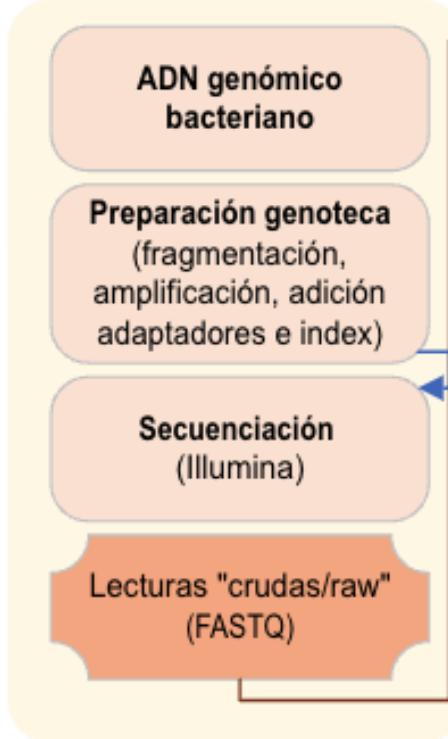
Actualmente, existen herramientas prediseñadas para el análisis de genomas completos a partir de sus ensamblajes como son FastANI (Jain et al., 2018), JSpecies (Richter y Rosselló-Móra, 2009) u OrthoANI (Lee et al., 2016).

A continuación, algunos vídeos donde se explica el cálculo de valores ANI y su interpretación.

<https://www.youtube.com/watch?v=zNsetoW7USc>

<https://www.youtube.com/watch?v=FDmjALmmpck>

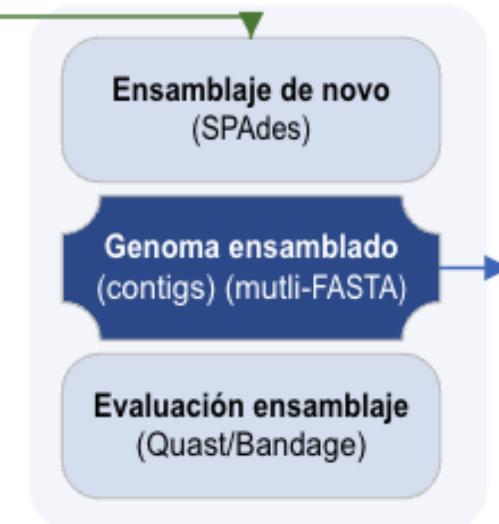
(1) Wet-Lab



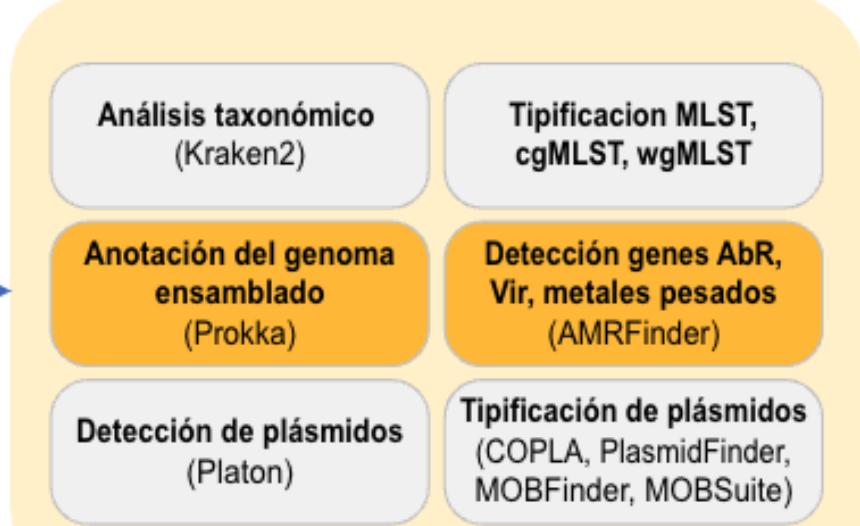
(2) Análisis calidad



(3) Ensamblaje



(4) Post-análisis



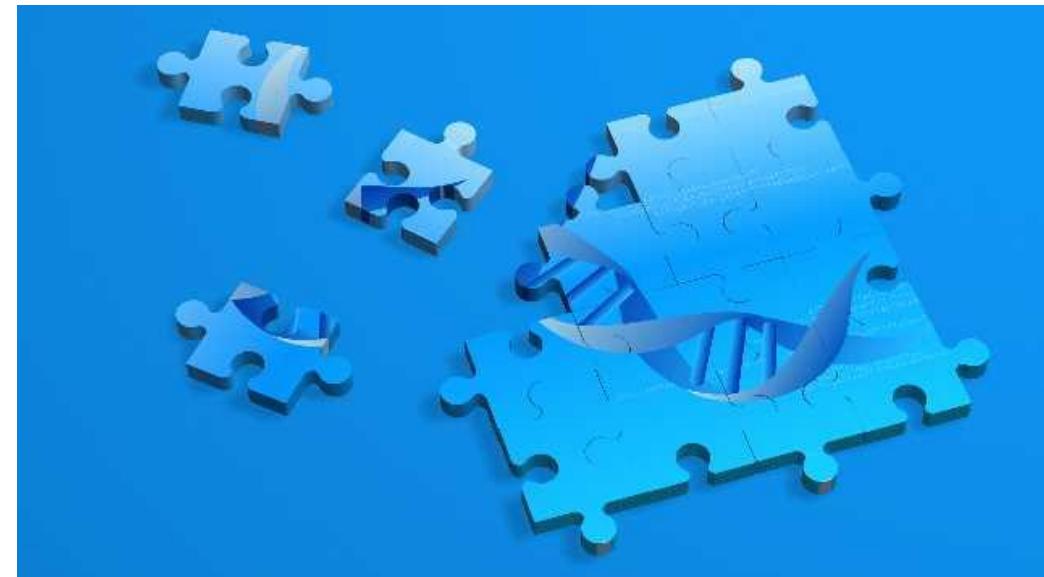
## 5.4.

Reconstrucción del genoma:  
*ensamblaje de novo*

El ensamblaje *de novo* es la técnica que consiste en unir fragmentos de secuencias para **reconstruir el genoma de un organismo, que ha sido previamente fragmentado y secuenciado**.

Hasta el momento, lo que habíamos visto en esta asignatura era la resecuenciación, la secuenciación basandonos en un molde o referencia sobre la cual reconstruir el genoma.

El ensamblaje *de novo* no toma referencia ninguna para reconstruir el genoma, lo que podría asimilarse a construir un puzzle de millones de pequeñas piezas sin tener la fotografía final como referencia.



Este proceso no resulta sencillo, debido fundamentalmente a los problemas que genera la fragmentación: **cuanto más fragmentado haya sido el genoma del organismo, más difícil es reconstruirlo**. Si volvemos a la analogía con un puzzle, imagináros que tenemos 30 millones de piezas que, a diferencia de los puzzles comerciales, sería difícil de encajar porque:

1. Pueden faltar piezas (debido a errores en secuenciación).
2. Existe un gran número de piezas iguales (regiones repetidas).
3. Algunas de las piezas contienen errores (errores de secuenciación).

Para paliar algunos de estos problemas, los ensambladores necesitan entre 5-10 copias de cada pieza para poder ensamblar, lo que sería tener una **cobertura media de entre 5 a 10x**.

El proceso de ensamblaje no está exento de **errores** y tiene varias explicaciones:

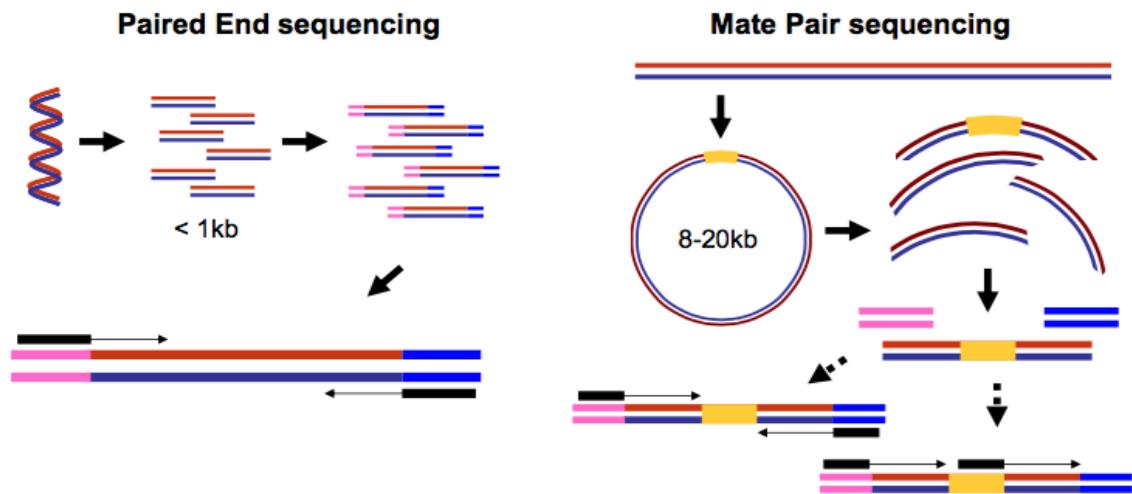
- Habitualmente **descartamos fragmentos que pensamos contienen errores**, en ocasiones de manera incorrecta, con lo que perdemos regiones secuenciadas.
- La unión de fragmentos en el sitio incorrecto o en orientación equivocada, formando **regiones quimera**. La única manera en la que podríamos paliar este tipo de problemas es tener lecturas muy largas y sin errores. Esto es parcialmente posible hoy en día, ya que las tecnologías de lecturas largas (PacBio y Oxford Nanopore) cada vez contienen menos errores, pero aún superan con creces los errores que comete Illumina\*.

Por eso, para concluir un ensamblaje con éxito: menor número de errores y maximizando la longitud ensamblada, aún requiere de ensamblajes híbridos entre lecturas largas y cortas.

\*Desde septiembre 2022 se encuentra disponible la química Q20+ para Nanopore (chips R10.4.1 + duplex reads), donde las tasas de error se han disminuido.  
Más información: <https://nanoporetech.com/q20plus-chemistry>

## 5.4.1. Genotecas para el ensamblado *de novo*

Tipo de librería	Ventajas	Inconvenientes	Plataforma
Lecturas cortas <i>single-end</i>	Aportan profundidad, lo que determina mayor fiabilidad y precisión. Hoy en día están en desuso.	No sirven para reconstruir grandes regiones del genoma. Propensas a artefactos de ensamblaje.	Illumina y SOLiD
Lecturas cortas emparejadas/ <i>paired-end</i>	Dan profundidad al ensamblaje, pero además es mucho más exacto al sortear los elementos repetidos de longitud menor a la del inserto entre las dos lecturas.	No se resuelven la mayor parte de las repeticiones, ya que los insertos suelen ser pequeños (< 500 pb).	Illumina y SOLiD
Lecturas cortas conjugadas ( <i>mate-pair</i> )	Muy usadas en conjunto con los dos tipos anteriores, ya que permiten reconstrucción de fragmentos mayores. Incorporan insertos de unos 10 kb.	No aportan profundidad al ensamblaje, son bastante costosas y no siempre se obtienen buenos resultados.	Illumina
Lecturas largas	Resuelven las repeticiones más largas, ya que las lecturas suelen ser de entre 10-50 kb-	Tasa de error de secuenciación muy alta. No aportan profundidad.	PacBio y Oxford Nanopore



## 5.4.2. Conceptos generales de ensamblaje

Un ensamblaje clásico suele estar compuesto por varios conjuntos de lecturas sueltas o emparejadas. La unión de lecturas sueltas (single-end) por solapamiento de una secuencia continua llamada unitig. Los unitig a su vez, pueden unirse entre ellos formando secuencias continuas llamadas **contigs**. Las secuencias repetidas, los polimorfismos, los fragmentos perdidos y los errores provocan el acortamiento de la longitud de los contigs resultantes.

Habitualmente utilizamos el término **contig** directamente, para hacer referencia a una secuencia continua. La diferencia entre unitig y contig radica en cómo se generan dentro del algoritmo de ensamblaje. Es una diferencia sutil: el unitig ha sido creado con la certeza de que es correcto porque es la única posibilidad, mientras que el contig se obtiene tras decidir entre varias opciones según el criterio del ensamblador.

## 5.4.2. Conceptos generales de ensamblaje

Por otro lado, el uso de **lecturas emparejadas o conjugadas (mate-pair)** es extremadamente útil en las últimas fases del ensamblaje. Cada par de estas lecturas se encuentra a una distancia conocida, que varía en función del protocolo de preparación de la genoteca empleado (200 pb hasta decenas de kilobases). Como las parejas de lecturas se generaron a partir del mismo fragmento de ADN, la formación de contigs se simplifica y nos permite unir estas estructuras en scaffolds. Los **scaffolds** son un conjunto de contigs ordenados entre sí, con huecos (gaps) de tamaño conocido entre ellos.

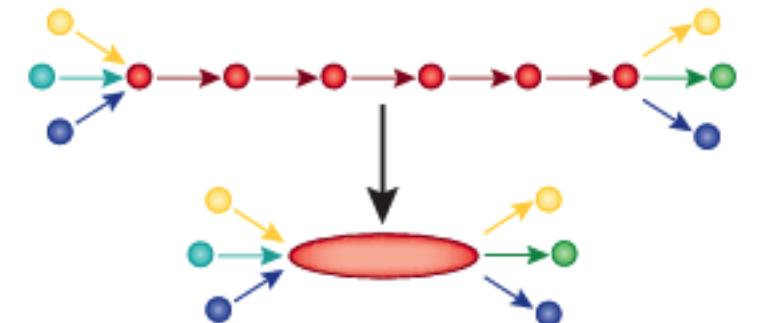
### 1. Fragment DNA and sequence



### 2. Find overlaps between reads



### 3. Assemble overlaps into contigs



### 4. Assemble contigs into scaffolds



Michael Schatz, Cold Spring Harbor

Genome assembly stitches together a genome from short sequenced pieces of DNA.

## 5.4.3. Tipos de algoritmos de ensamblaje

Existen tres tipos de algoritmos de ensamblaje de novo (Miller et al., 2010):

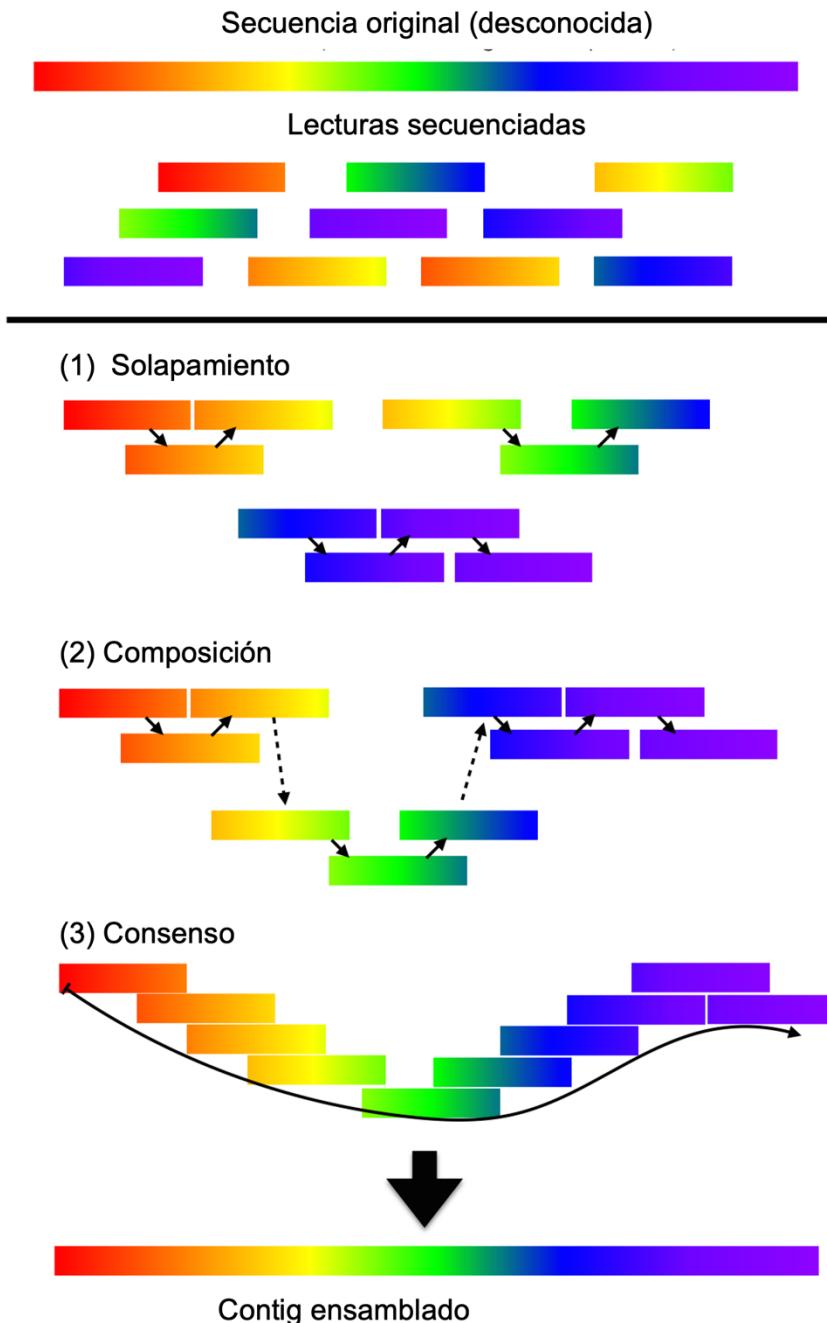
1. **Algoritmo de solapamiento-composición-consenso (OLC, overlap layout consensus)**, que fueron los primeros en utilizarse. Son muy intuitivos, pero **computacionalmente muy costosos**, por lo que no son apropiados para las lecturas brutas de NGS. Siguen teniendo su campo de aplicación, dada la enorme fiabilidad de sus resultados.
2. **Algoritmos “voraces” (greedy)**. Aparecieron con los primeros datos de NGS, cuando las lecturas eran muy cortas, de entre 25-35 pb, y no se podían aplicar los algoritmos OLC. Debido a su “voracidad”, **se comen, como mínimo, las secuencias repetitivas**. Hoy en día ya no se utilizan.
3. **Algoritmos basados en grafos de De Bruijn (DBG)**. Son los más utilizados hoy en día porque son muy eficaces computacionalmente.

Vamos a ahondar en los algoritmos OLC y De Bruijn, por ser los más comunes en ensamblaje de novo, incluyendo una breve reseña sobre teoría de Grafos.

En el manual tenéis descrita la teoría de grafos.

# Algoritmos OLC

1. **Solapamiento.** Se buscan solapamientos entre las lecturas comparándolas todas contra todas para saber qué lecturas solapan entre sí. El proceso es computacionalmente muy intensivo, puesto que se dispara el número de comparaciones con cada lectura que entra en juego ( $2n$ ). Lo habitual es buscar lecturas que solapen como mínimo 40-60 pb en sus extremos (por eso no eran útiles en las primeras lecturas obtenidas, que no llegaban a esta longitud). Un solapamiento considerado aceptable sería el perfecto (100 %), pero debido a polimorfismos y errores de secuenciación, se considera aceptable si ronda el 95-99 % de la región solapante. Por esto, parámetros como la longitud mínima del solapamiento y el porcentaje mínimo de identidad requerido en dichos solapamientos son dos parámetros configurables y de importancia en este paso. Tras la selección de los mejores solapamientos se genera un grafo (Figura 25-1). Originalmente, muchos de los programas OLC utilizaban versiones optimizadas del algoritmo de Smith-Waterman, para alineamientos locales, que dividía cada lectura en palabras, lo que sería equivalente a los k-meros en teoría de grafos.
2. **Composición.** El grafo resultante de los solapamientos no necesita incluir todas las secuencias de partida, sino tan solo un conjunto representativo (Figura 25-2). Los caminos extraídos de este grafo darán lugar a los contigs, creciendo con el número de lecturas. Puede volverse inmanejable y complicado. Cada uno de estos caminos en busca de los contigs se denomina camino de Hamilton o hamiltoniano.
3. **Consenso.** Una vez obtenidos los contigs, hay que incorporar las secuencias que no formaron parte del grafo para corregir los posibles errores que tengan las lecturas previamente seleccionadas. Esto se consigue con un alineamiento múltiple de los contigs y las lecturas de partida para establecer el consenso (Figura 25-3). Debemos ser cuidadosos y conscientes de que este paso puede generar secuencias químéricas que en realidad no existen.



Los algoritmos OLC son los más precisos y consiguen mejores resultados, pero solo pueden ser usados con lecturas largas y de gran calidad, básicamente restringiéndose a lecturas Sanger o pirosecuenciación 454 de Roche (discontinuada en 2016). No son algoritmos recomendables para secuencias Illumina por ser cortas, ni para PacBio u Oxford Nanopore, por la elevada tasa de error\*.

Como ejemplo de ensambladores basados en OLC están los clásicos CAP, Newbler y MIRA (Miller *et al.*, 2010).

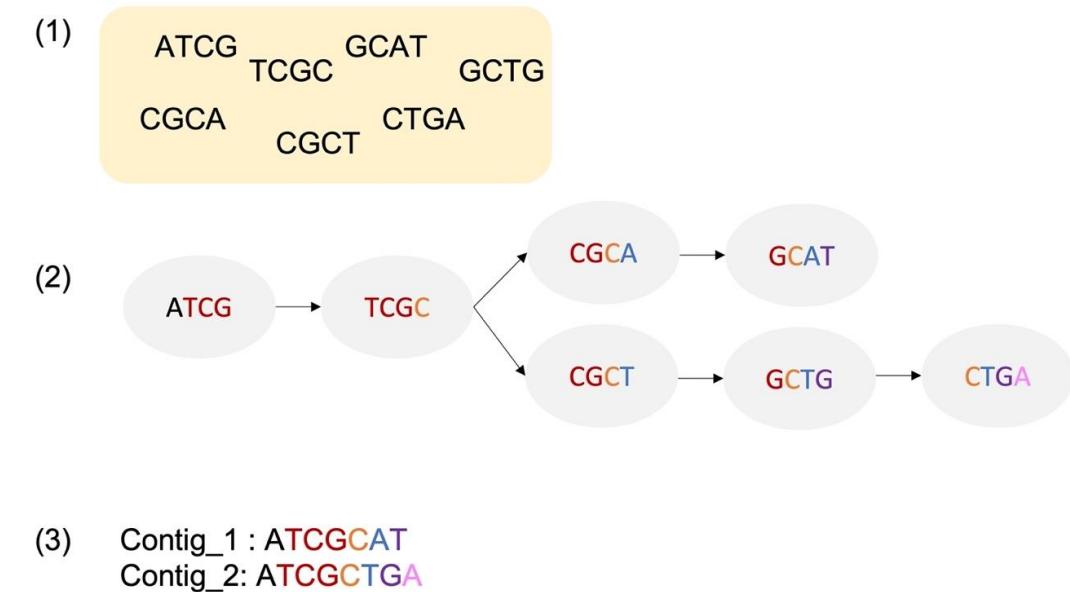
\*Desde septiembre 2022 se encuentra disponible la química Q20+ para Nanopore (chips R10.4.1 + duplex reads), donde las tasas de error se han disminuido. Más información: <https://nanoporetech.com/q20plus-chemistry>

# Algoritmos basados en grafos de De Bruijn (DBG)

Estos algoritmos surgieron para las lecturas cortas de las plataformas SOLiD e Illumina, pero hoy en día son de uso universal, por ser muy eficaces desde el punto de vista computacional.

Su eficacia computacional se debe a que se basan en grafos de *k*-meros que no requieren una comparación de todos contra todos, a diferencia de los algoritmos OLC. A cambio, no se mantienen las secuencias originales, lo que provoca que no sean tan precisos.

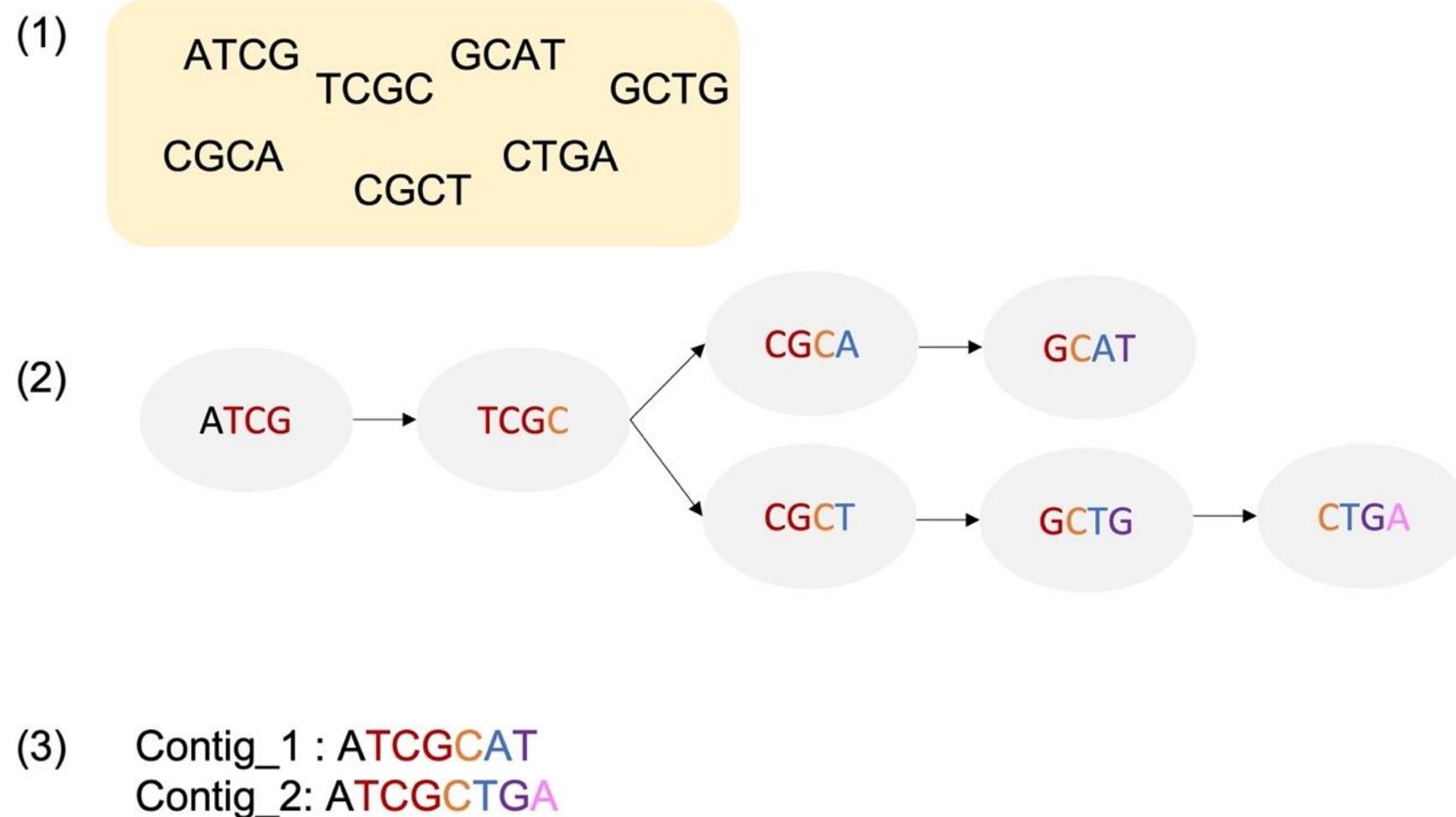
Otra diferencia fundamental es que los grafos de De Bruijn hay que resolverlos mediante caminos de Euler (estrategia euleriana) que consiste en buscar el camino que cubre todo el grafo y pasa por cada nodo una única vez. Los polimorfismos y los errores de secuenciación, así como las regiones repetidas, hacen que no se pueda encontrar un único camino y que el ensamblador se detenga en ciertas bifurcaciones.



## Algoritmos basados en grafos de De Bruijn (DBG)

Las etapas por las que pasa un ensamblador de este tipo son:

1. **Fragmentación.** Se fragmentan las lecturas en k-meros de longitud (k) constante y solapantes de nucleótido en nucleótido. Esto evita comparar todas las lecturas contra todas ellas. k suele oscilar entre 19 pb y la longitud total de la lectura. Cuanto más pequeño es k, más computación es necesaria; mientras que a mayor k-mero, más fiable es el solapamiento y menos sobrecarga computacional, siendo más sensibles a los errores. Hay que ser cautos porque muchos ensambladores admiten un margen pequeño de k-meros y algunos incluso tenemos que compilarlos para cambiar el único valor de k-mero que admiten.
2. **Construcción del grafo** en el que se conectan los k-meros que solapan k-1 nucleótidos entre sí. Al reducir a un k-mero todas las lecturas que lo contienen, se suprime la redundancia, con lo que el grafo se reduce y simplifica, volviéndose más manejable.
3. **Búsqueda de caminos eulerianos que reconstruyan la secuencia original.**



Los factores que complican el ensamblaje a partir de grafos basados en k-meros son:

- Las repeticiones directas generan bucles en los grafos de los k-meros, por lo que impiden una única reconstrucción de la secuencia original.
- El ADN tiene dos cadenas, por lo que la secuencia de una lectura seguro solapa con su complementaria. Para solventar este problema, los ensambladores incorporan en cada nodo enlaces para las dos cadenas, para impedir ensamblar la secuencia dos veces.
- Las repeticiones complejas (tándem, invertidas, imperfectas, etc.) generan estructuras en el grafo altamente ramificadas, complejas y difíciles de resolver, que solo se pueden solventar con k-meros superiores a la longitud de la repetición, y eso por ahora solo es posible con tecnología PacBio u Oxford Nanopore.
- Los errores de secuenciación se intentan soslayar con un preprocessamiento de las lecturas para eliminarlos y ponderarlos en los enlaces del grafo, en función del número de lecturas que los soportan.

Los ejemplos más conocidos de ensambladores de este tipo, utilizados actualmente son **Velvet** (Zerbino, 2010; Zerbino y Birney, 2008) y **SPAdes** (Bankevich et al., 2012).

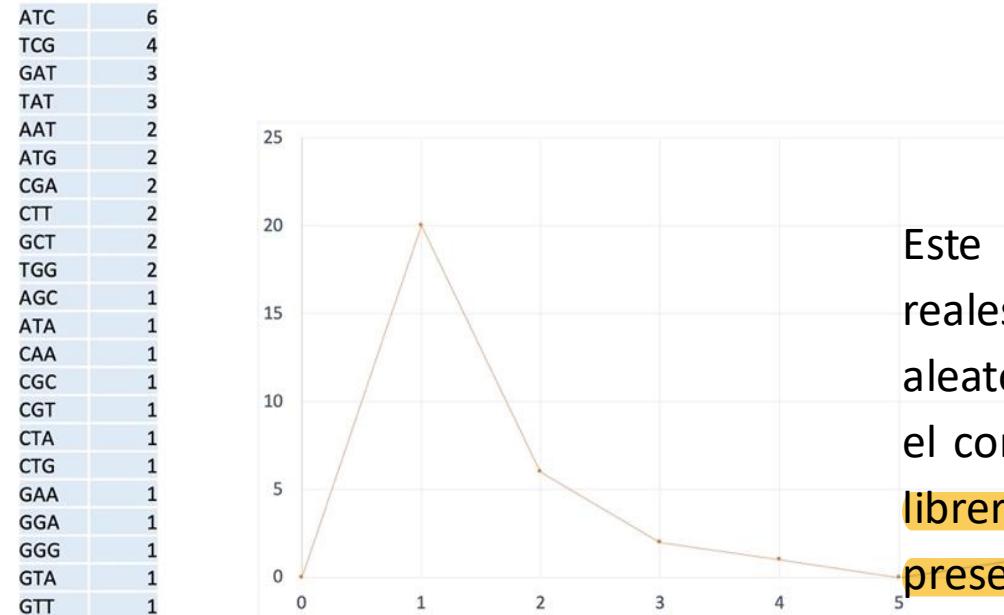
## 5.4.4. Evaluación previa al ensamblaje. Análisis de k-meros.

Previo al ensamblaje y posterior al análisis de calidad de las lecturas mediante FastQC y limpieza de adaptadores y regiones de baja calidad con programas como Trimmomatic, TrimGalore o FastP, podemos realizar un análisis de k-meros para **localizar posibles errores o contaminaciones de otros genomas**, vectores, adaptadores. Este es un paso opcional, que se encuentra incluido en algunas versiones de programas como FastQC y FastP.

Imaginemos una secuencia de nucleótidos de 50 pb que se ha generado en la secuenciación masiva, que vamos a dividir en k-meros de 4 nucleótidos (con lo que la longitud que utilizaremos de solapamiento será de  $k-1 = 3$  pb). Para ello, recorremos la secuencia con ventanas de longitud 4 desplazadas nucleótido a nucleótido, con lo que serán solapantes.

Vemos un ejemplo en la Figura 26. Cada k-mero se guarda en una tabla hash, junto con el número de veces que aparece ese k-mero en la secuencia. Si representamos el número de veces que aparece un k-mero en la secuencia, vemos que en este ejemplo tan sencillo, la mayoría de ellos aparecen una única vez, mientras que solo unos pocos aparecen dos o más veces.

ATCGCTATGTTGAATAGCTTATCGATGGGATCAATCGTATCGATCCTGG



Este tipo de curvas se mantiene en los datos reales, de lo que deducimos que la generación aleatoria de lecturas representará con fidelidad el contenido de la secuencia original. **Todas las librerías de un mismo genoma deberían presentar el mismo espectro de  $k$ -meros, y el genoma ensamblado también.** Cualquier desviación de esta situación nos ayudará a estimar la representatividad de las librerías y del ensamblaje con respecto a la secuencia original que queremos reconstruir.

## 5.4.4. Evaluación previa al ensamblaje. Análisis de k-meros.

Aunque existen varias herramientas bioinformáticas que nos proporcionan análisis de k-meros la más indicada para realizarlo es KAT.

Disponible en el siguiente enlace:

<https://github.com/TGAC/KAT>

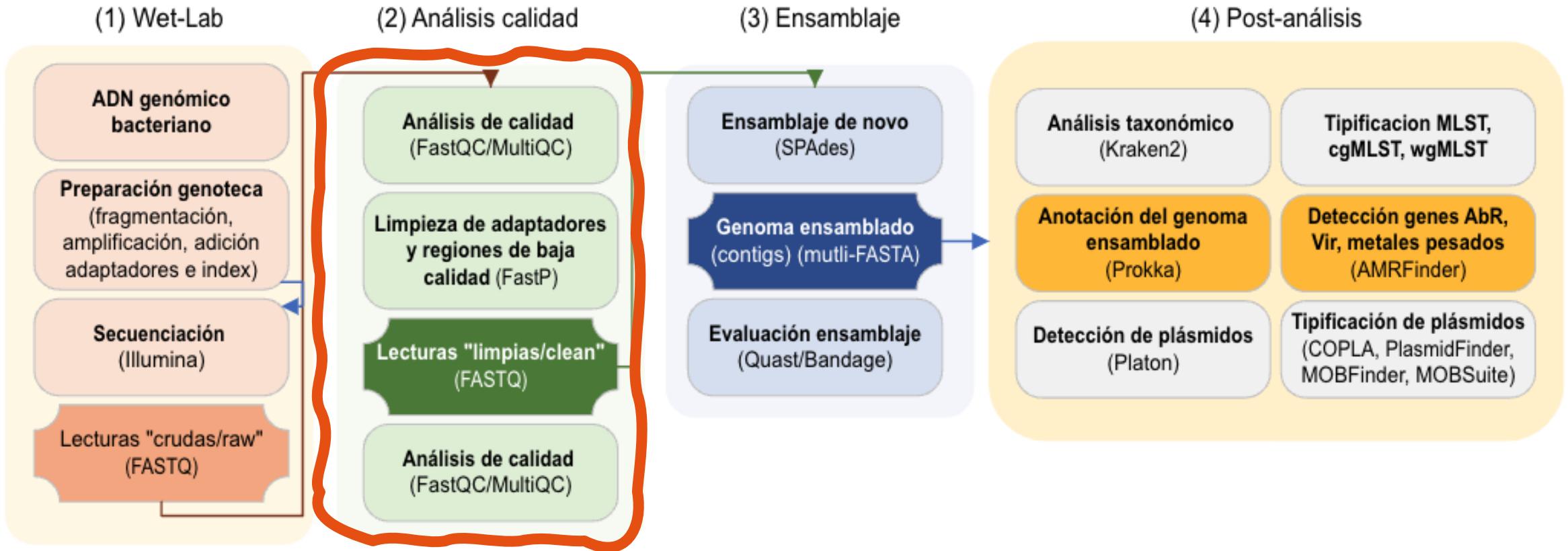
Con una documentación extensa en:

<https://kat.readthedocs.io/en/latest/>

Y DSK:

<https://github.com/GATB/dsk>

<https://gatb.inria.fr/software/dsk/>



# Tema 5 – Ejemplo 1 - Análisis y limpieza de calidad de las lecturas



## Tema 5 - Ejemplo 1 - Análisis y filtrado de calidad de las lecturas

El primer paso a realizar en cualquier análisis bioinformático es el análisis de calidad de las lecturas, así como su limpieza de calidad (filtrado de regiones de baja calidad, filtrado de adaptadores, regiones polyG o polyX...). En este caso vamos a analizar unas lecturas Illumina de un genoma bacteriano descargado de la base de datos pública Bioproject. Analizaremos la calidad de la secuencia con FastQC y MultiQC y filtraremos por calidad con FastP.

1) Instalar y activar el entorno Conda para genoma bacteriano y datos Illumina, disponible en Materiales y Recursos del aula virtual:

[04MBIF\\_bacteriano.yml](#)

```
mamba env create -f 04MBIF_bacteriano.yml
```

```
mamba activate 04MBIF_bacteriano
```

2) Descargar las secuencias de trabajo, disponibles en: [reads\\_illumina](#)

3) Realizar el análisis FastQC y MultiQC.

```
fastqc *.fastq.gz
```

```
multiqc --interactive .
```

4) responder a las siguientes preguntas

- a. ¿Cuántas lecturas tiene el genoma secuenciado?
- b. ¿Qué cobertura teórica tendría este genoma, sabiendo que es un genoma de E.coli (aprox. 5 Mb)?
- c. ¿Contienen estas secuencias adaptadores?
- d. ¿Cuál es la longitud de lectura?
- e. ¿Consideráis necesario hacer una limpieza previa al ensamblado?

5) Realizar la limpieza de adaptadores y regiones de baja calidad con FastP, utilizando el comando siguiente:

```
fastp -i CFSAN112772_S36_L001_R1_001.fastq.gz -I CFSAN112772_S36_L001_R2_001.fastq.gz -o out_1.fastq.gz -O out_2.fastq.gz --cut_tail 25 --cut_front 25 --cut_mean_quality 25 -l 151 -h out_FastP.html
```

6) Contesta a las siguientes preguntas sobre esta ejecución:

- a. ¿Qué significa cada uno de los parámetros utilizados?
- b. ¿Cuántas lecturas obtenemos tras el filtrado?
- c. ¿Qué porcentaje de bases Q20 y Q30?
- d. ¿Cuál es la tasa de duplicación?
- e. ¿Qué tamaño de inserto se ha detectado?
- f. ¿Cuál es el tamaño medio de las lecturas antes de limpieza? ¿y después?
- g. ¿Cuál es el contenido GC?

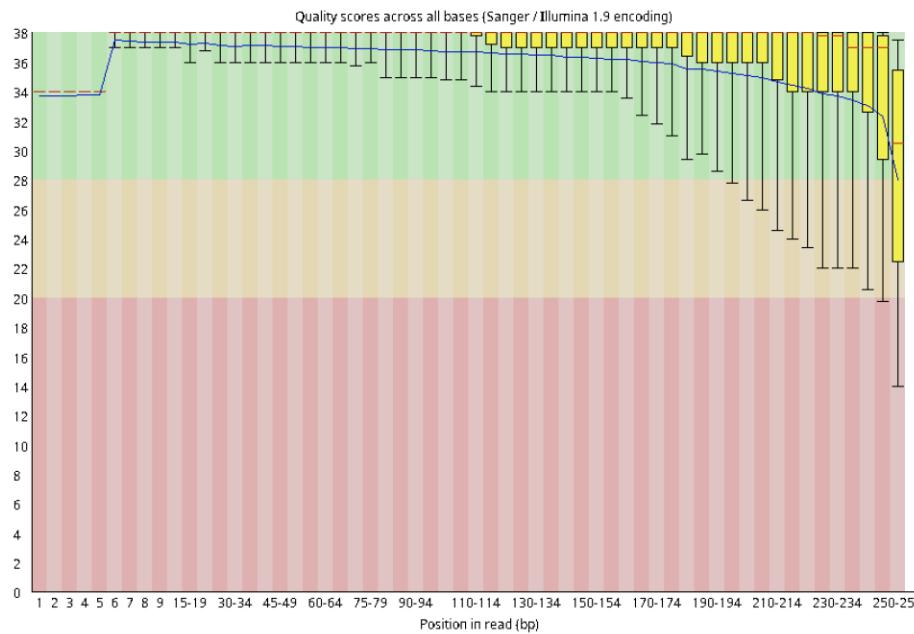
Al multiqc hay que darle el  
directorio, si es en el mismo que  
estás con poner un punto vale.

```
fastp -i CFSAN112772_S36_L001_R1_001.fastq.gz -I  
CFSAN112772_S36_L001_R2_001.fastq.gz -o out_1.fastq.gz -O out_2.fastq.gz --cut_tail 25 --  
cut_front 25 --cut_mean_quality 25 -l 151 --detect_adapter_for_pe -h report_fastp.htm
```

## Basic Statistics

Measure	Value
Filename	CFSAN112772_S36_L001_R1_001.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1058521
Sequences flagged as poor quality	0
Sequence length	35-251
%GC	50

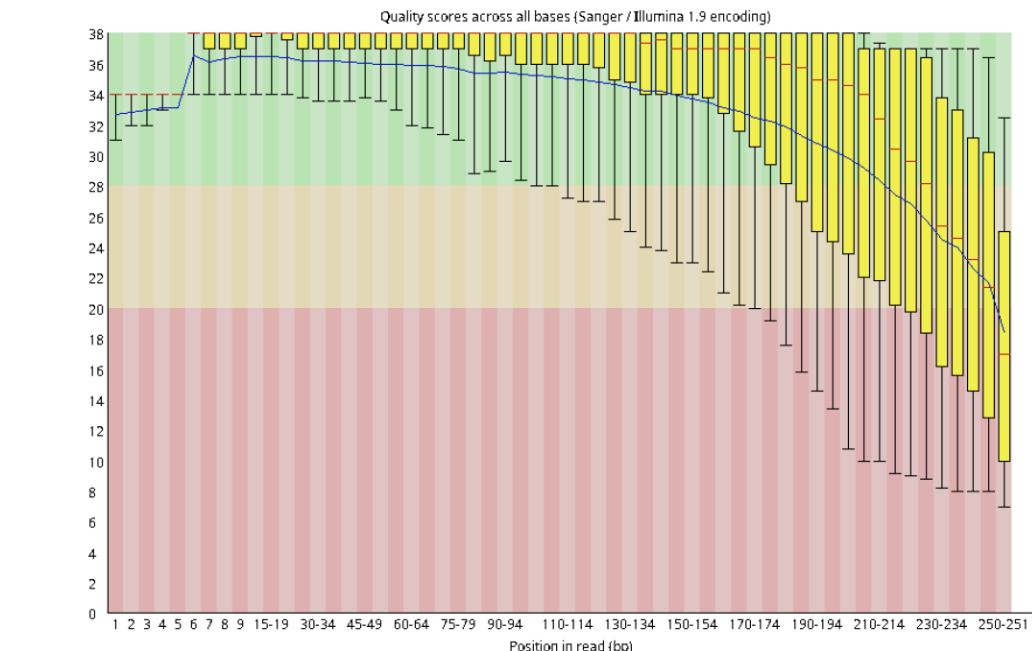
## Per base sequence quality



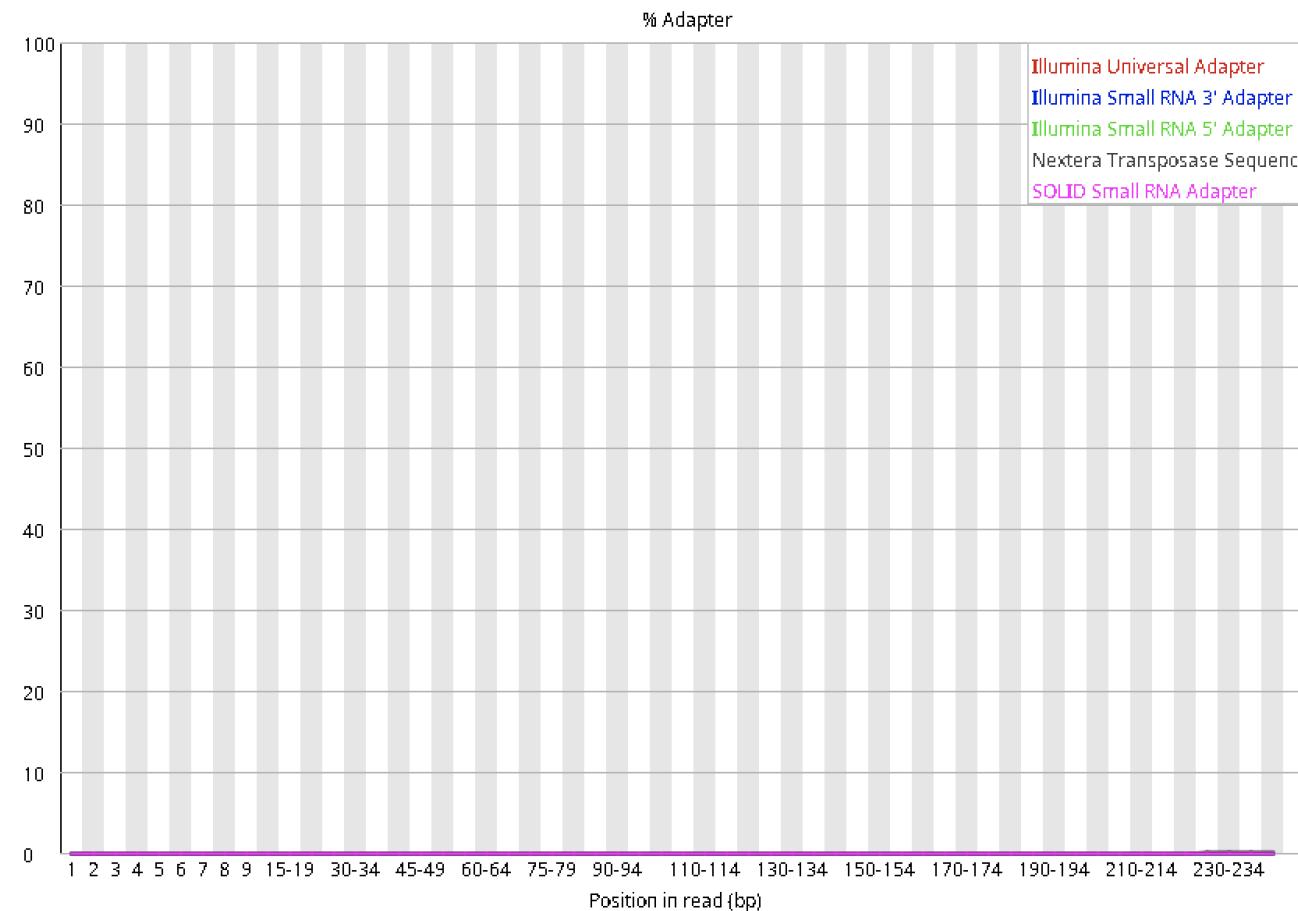
## Basic Statistics

Measure	Value
Filename	CFSAN112772_S36_L001_R2_001.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1058521
Sequences flagged as poor quality	0
Sequence length	35-251
%GC	50

## Per base sequence quality



## ✓ Adapter Content



```
Read1 before filtering:  
total reads: 1058521  
total bases: 243039818  
Q20 bases: 237305717(97.6407%)  
Q30 bases: 225588948(92.8197%)  
  
Read2 before filtering:  
total reads: 1058521  
total bases: 243824063  
Q20 bases: 221381579(90.7956%)  
Q30 bases: 195190734(80.0539%)  
  
Read1 after filtering:  
total reads: 970928  
total bases: 229658040  
Q20 bases: 225278323(98.0929%)  
Q30 bases: 214915832(93.5808%)  
  
Read2 after filtering:  
total reads: 970928  
total bases: 222239113  
Q20 bases: 208138295(93.6551%)  
Q30 bases: 185710696(83.5635%)  
  
Filtering result:  
reads passed filter: 1941856  
reads failed due to low quality: 24762  
reads failed due to too many N: 1698  
reads failed due to too short: 148726  
reads with adapter trimmed: 31888  
bases trimmed due to adapters: 365799  
  
Duplication rate: 0.238068%  
  
Insert size peak (evaluated by paired-end reads): 250  
  
JSON report: fastp.json  
HTML report: out_FastP.html  
  
fastp -i CFSAN112772_S36_L001_R1_001.fastq.gz -I CFSAN112772_S36_L001_R2_001.fastq.gz -o out_1.fastq.gz -O out_2.fastq.gz --cut_tail 25 --cut_front 25 --cut_mean_quality 25 -l 151 -h out_FastP.html  
fastp v0.23.2, time used: 33 seconds
```

# fastp report

## Summary

### General

<b>fastp version:</b>	0.23.2 ( <a href="https://github.com/OpenGene/fastp">https://github.com/OpenGene/fastp</a> )
<b>sequencing:</b>	paired end (251 cycles + 251 cycles)
<b>mean length before filtering:</b>	229bp, 230bp
<b>mean length after filtering:</b>	236bp, 228bp
<b>duplication rate:</b>	0.238068%
<b>Insert size peak:</b>	250

### Before filtering

<b>total reads:</b>	2.117042 M
<b>total bases:</b>	486.863881 M
<b>Q20 bases:</b>	458.687296 M (94.212636%)
<b>Q30 bases:</b>	420.779682 M (86.426555%)
<b>GC content:</b>	50.606693%

### After filtering

<b>total reads:</b>	1.941856 M
<b>total bases:</b>	451.897153 M
<b>Q20 bases:</b>	433.416618 M (95.910456%)
<b>Q30 bases:</b>	400.626528 M (88.654360%)
<b>GC content:</b>	50.538614%

### Filtering result

<b>reads passed filters:</b>	1.941856 M (91.724963%)
<b>reads with low quality:</b>	24.762000 K (1.169651%)
<b>reads with too many N:</b>	1.698000 K (0.080206%)
<b>reads too short:</b>	148.726000 K (7.025179%)

## Adapters

### Adapter or bad ligation of read1

The input has little adapter percentage (~0.022812%), probably it's trimmed before.

Sequence	Occurrences
C	13
CT	20
CTGTC	30
other adapter sequences	1041

### Adapter or bad ligation of read2

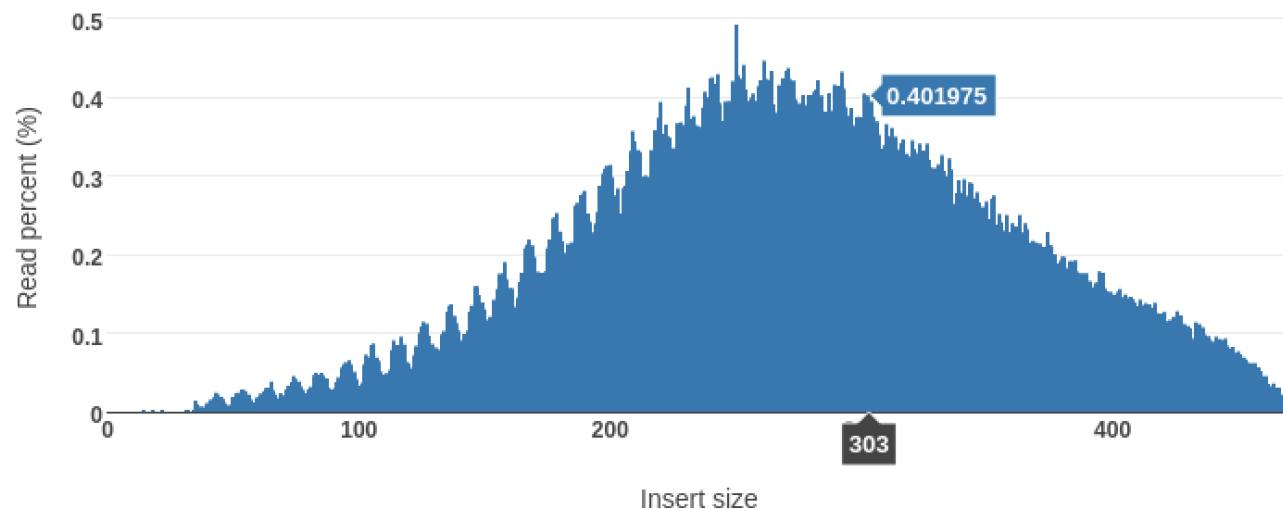
The input has little adapter percentage (~0.127455%), probably it's trimmed before.

Sequence	Occurrences
C	836
CT	497
CTG	305
CTGTC	415
CTGTCT	182
CTGTCCTC	170
CTGTCCTCT	239
CTGTCCTCTT	244
other adapter sequences	12647

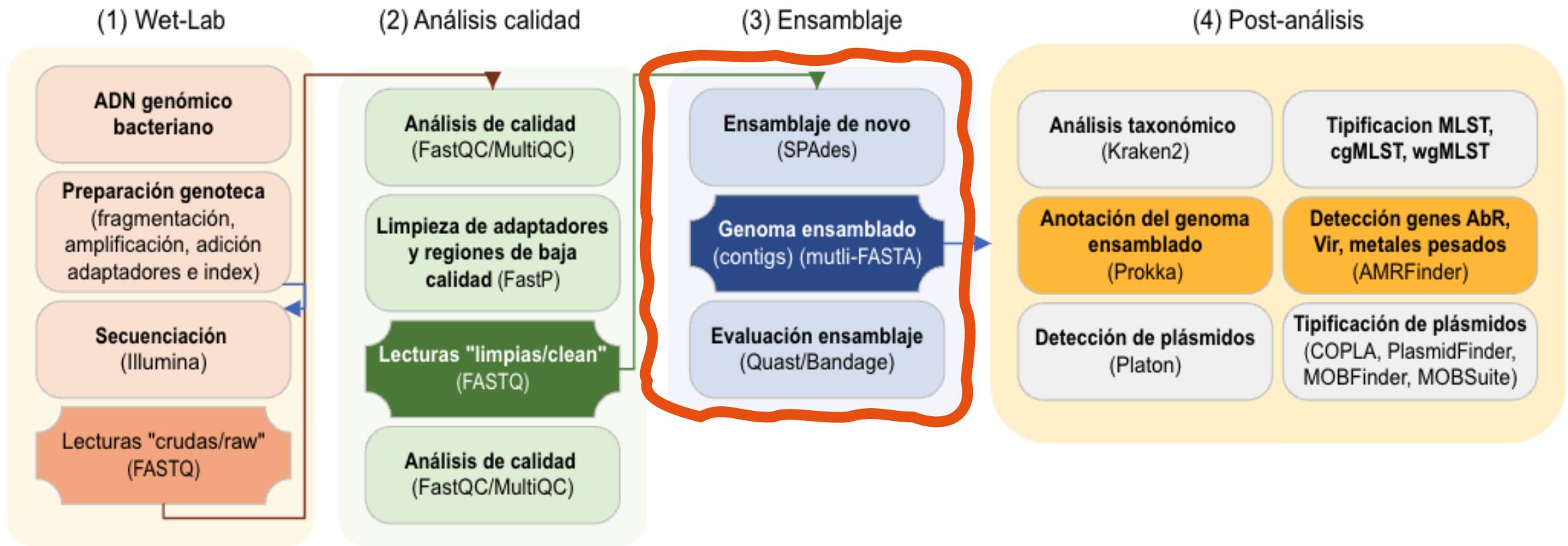
## Insert size estimation



Insert size distribution (11.583295% reads are with unknown length)



This estimation is based on paired-end overlap analysis, and there are 11.583295% reads found not overlapped.  
The nonoverlapped read pairs may have insert size <30 or >472, or contain too much sequencing errors to be detected as overlapped.



## 5.4.5. Ensamblaje de secuencias cortas con SPAdes

En este apartado vamos a ver cómo realizar el ensamblaje de unas lecturas que ya han sido evaluadas en calidad. Para ello utilizaremos el ensamblador SPAdes (Bankevich et al., 2012; Prjibelski et al., 2020), por ser uno de los más utilizados para genomas bacterianos. Siempre es deseable, antes de lanzarse a utilizar un ensamblador, echar un vistazo a las evaluaciones publicadas donde se compara el rendimiento y resultado de cada uno de los ensambladores, especialmente en genomas que sean similares al nuestro.

En el caso de **SPAdes** (<https://cab.spbu.ru/software/spades/>), se trata de un kit de **ensambladores del tipo de De Bruijn**, donde se albergan varios protocolos, que han sido descritos en (Prjibelski et al., 2020), y que contemplan los siguientes, aunque se encuentra en constante evolución:

**SPAdes**: ensamblador general para genomas (el que utilizaremos).

**metaSPAdes**: para metagenomas.

**plasmidSPAdes**: extracción y ensamblaje de plásmidos en secuencias de genoma completo.

**metaplasmidSPAdes**: extracción y ensamblaje de plásmidos en metagenomas.

**rnaSPAdes**: ensamblaje de transcriptomas *de novo* a partir de datos de RNAseq.

**biosyntheticSPAdes**: ensamblaje de clústeres de genes biosintéticos a partir de lecturas pareadas.

**coronaSPAdes**: ensamblaje *de novo* de SARS-CoV-2.

**rnaviralSPAdes**: ensamblaje *de novo* para sets de datos de ARN viral (transcriptoma, metatranscriptoma y metaviroma).

**metaviralSPAdes**: ensamblaje de metaviroma.

## 5.4.5. Ensamblaje de secuencias cortas con SPAdes

Las versiones más actuales permiten el uso de lecturas Illumina o IonTorrent, en solitario o combinadas con lecturas PacBio, Oxford Nanopore o Sanger, para realizar ensamblajes híbridos, así como *contigs* adicionales que pueden ser utilizados como lecturas largas. No se recomienda utilizar lecturas largas en solitario\*, sino en combinación con las cortas.

Asimismo, este ensamblador puede utilizar lecturas pareadas, no pareadas y conjugadas, de manera agrupada o sencilla.

En todo caso, SPAdes fue **diseñado para genomas pequeños**, como son genomas bacterianos, fúngicos o virus; no se recomienda su aplicación en genomas grandes.

Además de sus distintas formas de ensamblaje, en función del tipo de muestra del que provengan las lecturas, se proporcionan *scripts* para el contaje de *k*-meros (*spades-kmercounter*), construcción del gráfico de ensamblaje (*spades-gbuilder*) y el alineamiento de lecturas largas a un grafo (*spades-gmapper*).

\*Hoy en día, debido a la disminución de la tasa de error de lecturas de este tipo, ya utilizamos sin problemas lecturas largas en solitario.

## 5.4.5. Ensamblaje de secuencias cortas con SPAdes

El **procedimiento** que sigue SPAdes consta de tres pasos:

**Corrección de errores.** Este **paso es opcional** y **puede realizarse fuera del protocolo SPAdes**, con otras herramientas. SPAdes proporciona dos de ellas integradas: BayesHammer y IonHammer, para lecturas Illumina y IonTorrent, respectivamente. Este es un proceso largo (aprox. 25-30 minutos para un genoma bacteriano de *E. coli*, con 25-30 M de lecturas pareadas de 100 pb).

**Ensamblaje.** Este segundo paso es un proceso iterativo, donde **se seleccionan unos valores de *k*-mero automáticamente basados en la longitud de la lectura y tipo de lecturas**. De este proceso se obtienen un conjunto de *contigs* y de *scaffolds*.

**Corrección de *mismatches*.** Esta herramienta **mejora la detección de *mismatches* e *indels* cortos en los *contigs* y *scaffolds* resultantes**. Para ello **utiliza la herramienta BWA**. Aunque esta opción está desactivada por defecto, los autores recomiendan utilizarla.

## 5.4.5. Ensamblaje de secuencias cortas con SPAdes

Recordad visitar el manual de SPAdes en el siguiente enlace o bien, en la terminal tecleando spades.py -h.

<https://cab.spbu.ru/files/release3.15.3/manual.html>

A continuación, vamos a ver algunos de los parámetros más sensibles que se debe tener en cuenta cuando utilicemos este programa.

- **Opciones básicas.** En opciones básicas es obligatorio determinar una carpeta de salida para los archivos (-o output\_dir). Opcionalmente, podremos seleccionar alguno de los protocolos “especiales” de SPAdes para el ensamblaje de metagenomas, viroma, ARN o plásmidos, entre otros.
- **Datos de entrada.** Dada la diversidad de datos de entrada que podemos utilizar, SPAdes tiene una larga lista de comandos opcionales en este punto. Para un conjunto de lecturas pareadas Illumina, el método más estándar y sencillo, los indicaremos como -1 read\_1.fastq.gz y -2 read\_2.fastq.gz. Las lecturas de ensambladores como PacBio o Nanopore se indican como –pacbio y –nanopore. Adicionalmente, si tenemos un conjunto de contigs ensamblados de un genoma similar (referencia) o del mismo, de ensamblajes anteriores, podemos utilizarlos como “molde” con la opción –trusted-contigs.

## 5.4.5. Ensamblaje de secuencias cortas con SPAdes

- **Opciones adicionales:**

- **Corrección de errores.** Como se ha comentado anteriormente, se puede realizar un primer paso de corrección de errores. Por defecto está habilitado en el protocolo, pero si queremos realizar solo la corrección, sin realizar ensamblaje utilizaremos --only-error-correction; o si por el contrario queremos deshabilitar la corrección de errores y solo ensamblar utilizaremos –only-assembler.
- **Minimizar mismatches e indels.** Asimismo, la minimización de mismatches e indels cortos sobre los contigs y scaffolds finales debe ser habilitada manualmente con la opción –careful.
- **Tamaño de k-mero.** La opción -k, seguida de uno o varios números impares separados por comas, nos permite elegir el tamaño de k-mero. Por defecto, SPAdes hace una selección automática de estos valores, utilizando como dato la longitud de la lectura, con un número máximo de 128 nucleótidos. Este programa no permite k-meros superiores a este tamaño, aunque en la compilación del programa puede modificarse. Sin embargo, los desarrolladores no lo recomiendan. Este programa determina automáticamente de entre los k-meros seleccionados la mejor opción. **Este paso es de extrema importancia.** La determinación del mejor k-mero define el ensamblaje. Es imposible saber de antemano el valor idóneo, ya que dpende de la calidad, longitud y tipo de lecturas, así como del organismo que estemos secuenciando. Para su elección, lo mejor es ensamblar con varios k-meros y evaluar. Debemos tener en cuenta que a mayor valor k, menos contigs generamos y más largos, pero nos arriesgamos a perder una parte importante del genoma y a forzar ensamblajes híbridos (quimera) no reales. Por otra parte, una k baja nos lleva a ensamblajes con más contigs y más cortos, con una mayor representación del genoma, pero más errores.
- **Cobertura mínima (--cov-cutoff):** con este parámetro, por defecto desactivado en el protocolo, se puede controlar la cobertura mínima que debe tener un unitig para ser considerado como tal.



## Tema 5 - Ejemplo 2 - Ensamblaje de novo



Una vez que tenemos las lecturas Illumina limpias, vamos a pasar a ensamblarlas (reconstruir el puzzle), para lo que utilizaremos el ensamblador SPAdes. Evaluaremos distintos k-meros y analizaremos los archivos de salida que nos proporciona el programa.

- 1) Localizar las lecturas limpias del ejemplo 1, obtenidas con FastP. Si no las tienes, puedes descargarlas de aquí: [FastP](#)
- 2) Ejecutar el ensamblador utilizando el modo "careful" y los kmeros en modo automático.

```
spades.py -1 out_1.clean.fastq.gz -2 out_2.clean.fastq.gz --careful -k auto -o spades_out
```

Nota: este proceso es bastante largo, así que tenéis las soluciones en este link: [spades\\_out.zip](#)

Repasa la carpeta de salida con los archivos obtenidos. ¿Qué kmeros se han testado? ¿Cuántos contigs se han obtenido? ¿Cuántos scaffolds?

Una vez finalizado el proceso (aprox. 60 minutos) podemos encontrar los siguientes archivos en la carpeta de salida (más información en <https://cab.spbu.ru/files/release3.12.0/manual.html>):

- spades.log y params.txt: archivos que nos muestran todo el proceso realizado y los parámetros utilizados. En él podemos encontrar si han existido errores, así como los pasos realizados y las versiones de los programas y dependencias.
- Lecturas corregidas: se encuentran en la carpeta corrected. En ellas están las lecturas corregidas de errores pareadas y no pareadas.
- Carpetas K21, K33, K55, K77, K99 y K127: corresponde al proceso para cada k-mero seleccionado. En su interior están los contigs ensamblados y los grafos del ensamblaje.
- Archivo contigs.fasta: contigs ensamblados con el k-mero seleccionado como óptimo por el programa.
- Archivo before\_rr.fasta: archivo de contigs antes de resolver repeticiones. No se recomienda trabajar con él.
- Archivo scaffolds.fasta: scaffolds reconstruidos con el k-mero seleccionado como óptimo por el programa. Son los recomendados para trabajar posteriormente.
- Archivo assembly\_graph.fastg, assembly\_graph\_with\_scaffolds.gfa: grafos de ensamblaje. Pueden ser visualizados en herramientas como Bandage (<http://rrwick.github.io/Bandage/>).
- Archivos de “path” en el grafo de ensamblaje correspondientes a los contigs y los scaffolds: contigs.paths y scaffolds.paths.

```
* Corrected reads are in /home/maria.detoro/04MBIF/Tema5_bacterianos/spades_out/corrected/
* Assembled contigs are in /home/maria.detoro/04MBIF/Tema5_bacterianos/spades_out/contigs.fasta
* Assembled scaffolds are in /home/maria.detoro/04MBIF/Tema5_bacterianos/spades_out/scaffolds.fasta
* Paths in the assembly graph corresponding to the contigs are in /home/maria.detoro/04MBIF/Tema5_bacterianos/spades_out/contigs.paths
* Paths in the assembly graph corresponding to the scaffolds are in /home/maria.detoro/04MBIF/Tema5_bacterianos/spades_out/scaffolds.paths
* Assembly graph is in /home/maria.detoro/04MBIF/Tema5_bacterianos/spades_out/assembly_graph.fastg
* Assembly graph in GFA format is in /home/maria.detoro/04MBIF/Tema5_bacterianos/spades_out/assembly_graph_with_scaffolds.gfa

===== SPAdes pipeline finished.
```

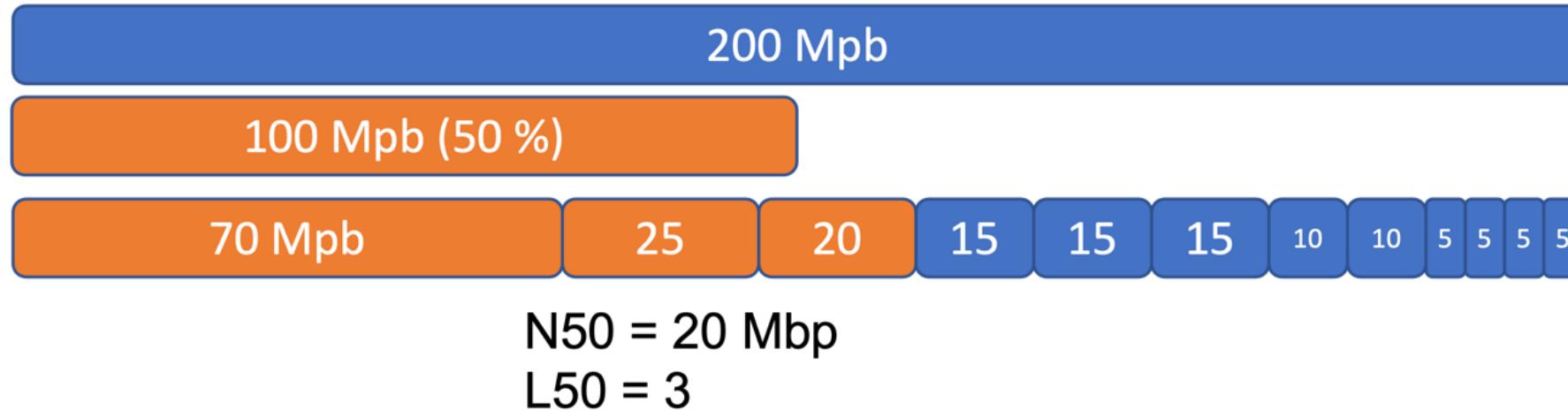
SPAdes log can be found here: /home/maria.detoro/04MBIF/Tema5\_bacterianos/spades\_out/spades.log

```
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8i1nqzd50ra spades_out]$ ll -h
total 43M
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 5.9M Dec 18 19:16 assembly_graph_after_simplification.gfa
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 13M Dec 18 19:16 assembly_graph.fasta
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 6.0M Dec 18 19:16 assembly_graph_with_scaffolds.gfa
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 6.0M Dec 18 19:16 before_rr.fasta
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 5.9M Dec 18 19:21 contigs.fasta
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 101K Dec 18 19:16 contigs.paths
drwxr-xr-x 3 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 18:52 corrected
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 86 Dec 18 17:59 dataset.info
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 230 Dec 18 17:59 input_dataset.yaml
drwxr-xr-x 4 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 19:16 K127
drwxr-xr-x 4 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 18:54 K21
drwxr-xr-x 4 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 18:57 K33
drwxr-xr-x 4 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 19:02 K55
drwxr-xr-x 4 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 19:07 K77
drwxr-xr-x 4 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 19:11 K99
drwxr-xr-x 2 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 19:25 misc
drwxr-xr-x 4 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 17:59 mismatch_corrector
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 1.6K Dec 18 17:59 params.txt
drwxr-xr-x 2 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 19:25 pipeline_state
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 5.2K Dec 18 17:59 run_spades.sh
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 8.1K Dec 18 17:59 run_spades.yaml
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 5.9M Dec 18 19:25 scaffolds.fasta
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 101K Dec 18 19:16 scaffolds.paths
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 240K Dec 18 19:25 spades.log
drwxr-xr-x 2 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 19:25 tmp
drwxr-xr-x 2 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 19:25 tmp
```

## 5.4.6. Evaluación de la calidad del ensamblaje.

Una vez realizado el ensamblaje debemos analizar la calidad de dicho ensamblaje en función de dos criterios:

- **Continuidad:** para valorar la fragmentación del ensamblaje.
- **Contenido:** se estudia la cantidad de información de las lecturas originales que no se ha incorporado al ensamblaje.



Nota. Si tenemos 12 *contigs* de longitudes (Mpb) 70, 25, 20, 15, 15, 15, 10, 10, 5, 5, 5 y 5, para calcular N50 primero los ordenaremos de mayor a menor, calculamos su suma (200 Mpb). Si revisamos qué *contig* sería el que abarca el 50 % del tamaño ensamblado (100 Mpb), sería 20 ( $70 + 25 + 20 = 115 \text{ Mpb}$ ), por tanto  $N50 = 20$ .

## 5.4.6. Evaluación de la calidad del ensamblaje.

Según la continuidad del ensamblaje, mediante la cual evaluamos la fragmentación, podemos determinar:

- **Número de bases del ensamblaje.** Aunque sea una métrica muy sencilla, nos da la estimación de cuánto hemos conseguido ensamblar, sobre todo si conocemos la longitud aproximada del organismo de referencia. Una desviación muy por encima de la longitud que esperamos, puede indicar contaminación o bien secuencias inesperadas (como por ejemplo fagos o plásmidos). En cambio, una desviación muy por debajo del número esperado nos indicará que una parte del genoma se ha quedado sin ensamblar.
- **Número de contigs/scaffolds.** Se suele utilizar este número para describir un ensamblaje. Un número muy elevado nos indicará que el ensamblaje está muy fragmentado y que será difícil trabajar con él. Sin embargo, forzar un ensamblaje a que tenga bajo número puede ser contraproducente, ya que podríamos estar forzando a la generación de secuencias quimera. Quizás lo más importante no sea el número, sino la longitud de dichos contigs/scaffolds. No es lo mismo trabajar con 1000 contigs de tamaño medio 500 pb, que hacerlo con 1000 contigs donde una parte de ellos superen, por ejemplo, los 5000 pb mientras que otros tantos sean pequeños y, por tanto, no nos aporten demasiada información.
- **Valor N50.** Esta métrica es la más utilizada porque es poco sensible a los valores extremos. Nos indica el tamaño del contig de forma que el 50 % del genoma ensamblado está soportado por bloques de este tamaño o mayores. Para calcularlo, se ordenan de mayor a menor longitud los contigs del ensamblaje, repitiendo cada elemento en la distribución tantas veces como sea su valor. La Figura 27 muestra un ejemplo de su cálculo.
- **Valores N20, N60, N90.** En la misma idea que la métrica N50, pero para los percentiles 20 %, 60 % y 90 %.
- **Valor L50.** Número de contigs que comprenden el 50 % del genoma ensamblado.

## 5.4.6. Evaluación de la calidad del ensamblaje.

Si atendemos al contenido podemos evaluar el ensamblaje en función de la cantidad de información de las lecturas originales que no se han incorporado al mismo. Existen dos estrategias:

- **Mapeo de las lecturas.** Se remapean las lecturas de partida sobre el ensamblaje considerado definitivo. La situación ideal sería que todas las lecturas mapeasen sin errores, sin embargo, siempre existen lecturas que no se mapean debido a errores. Esta cantidad nos da una idea aproximada.
- **Evaluación del espectro de k-meros.** Para ello se comparan las secuencias incorporadas en el ensamblaje, las totales, y las que no se mapean. Se puede evaluar su contenido en k-meros de ambos grupos, de manera que cuanto menos concuerden, más información se estará perdiendo.

La evaluación de los ensamblajes viene dada en muchos casos por el propio ensamblador, que muestra sus estadísticas. En el caso de SPAdes, esto no es así, de manera que podemos utilizar otro programa, como GAGE (Salzberg et al., 2012), Quast (Gurevich et al., 2013), CheckM (Parks et al., 2015), SQUAT (Yang et al., 2019) o GenomeQC (Manchanda et al., 2020), para evaluar todos estos parámetros.

Una buena guía para comprender algunos de estos conceptos es el artículo Do it yourself guide to genome assembly, de Wajid y Serpedin (2016):

<https://academic.oup.com/bfg/article/15/1/1/1741842>



## Tema 5 - Ejemplo 3 - Evaluación del ensamblaje



Una vez obtenido el ensamblaje con SPAdes (Ejemplo 2), podemos evaluar la bondad del ensamblaje. Para ello necesitaremos los contigs y scaffolds obtenidos, así como las lecturas originales. Utilizaremos el programa Quast, instalado en el environment de trabajo.

### 1) Ejecutar Quast:

```
quast.py -o quast_result -m 0 -t 2 --k-mer-size 127 --circos --pe1 ../out_1.clean.fastq.gz --pe2  
../out_2.clean.fastq.gz contigs.fasta scaffolds.fasta
```

### 2) Analicemos el resultado, a partir del archivo HTML obtenido:

- a. ¿cuántos contigs y cuántos scaffolds hemos obtenido? De éstos, ¿cuántos son mayores de 1 Kb?
- b. ¿Cuál es el tamaño del contig más largo?
- c. ¿Cuál es el total de nucleótidos ensamblados? ¿y en contigs mayores de 1 Kpb?
- d. ¿Cuál es el valor de N50? ¿L50?
- e. ¿Cuál es el contenido GC?
- f. ¿Cuántas N se encuentran por cada 100 Kpb?

Resultado: [Tema5\\_Ejemplo3\\_Quast](#)

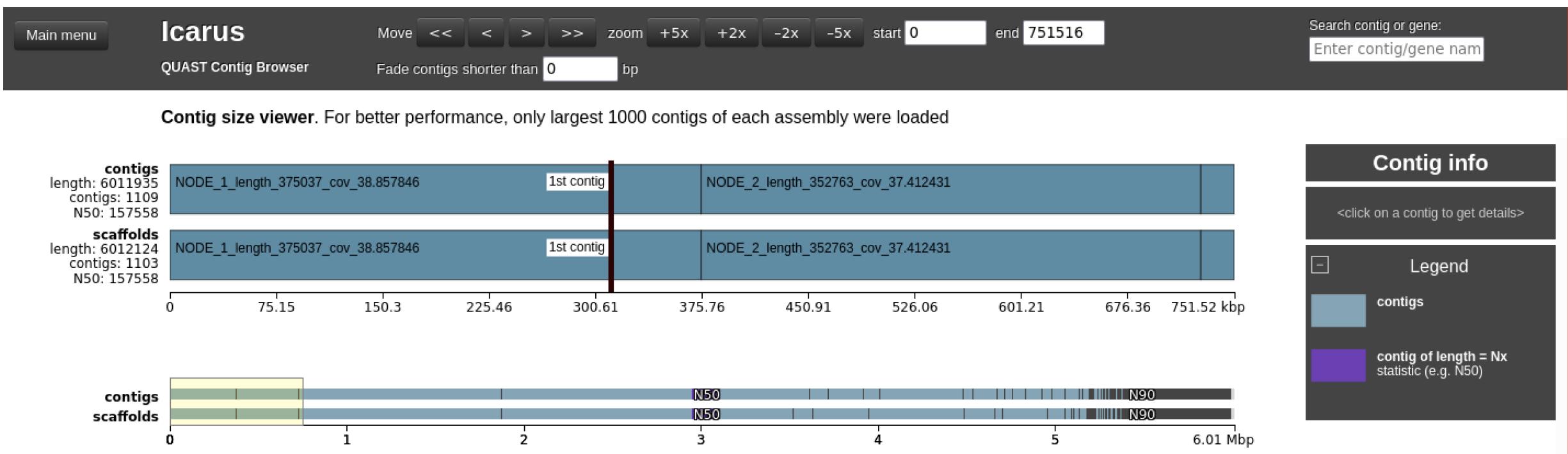
```
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8i1nqzd50ra quast_result]$ ll -h
total 584K
drwxr-xr-x 2 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 20:18 basic_stats
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 53K Dec 18 20:18 icarus.html
drwxr-xr-x 2 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 20:18 icarus_viewers
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 11K Dec 18 20:18 quast.log
drwxr-xr-x 2 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 4.0K Dec 18 20:18 reads_stats
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 433K Dec 18 20:18 report.html
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 42K Dec 18 20:18 report.pdf
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 1.8K Dec 18 20:18 report.tex
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 847 Dec 18 20:18 report.tsv
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 1.5K Dec 18 20:18 report.txt
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 1.5K Dec 18 20:18 transposed_report.tex
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\domain users 847 Dec 18 20:18 transposed_report.tsv
-rw-r--r-- 1 UNIVERSIDADVIU\maria.detoro UNIVERSIDADVIU\do 18 December 2022, Sunday, 20:18:38
```

[View in Icarus contig browser](#)

All statistics are based on contigs of size  $\geq 0$  bp, unless otherwise noted (e.g., "# contigs ( $\geq 0$  bp)" and "Total length ( $\geq 0$  bp)" include all contigs).

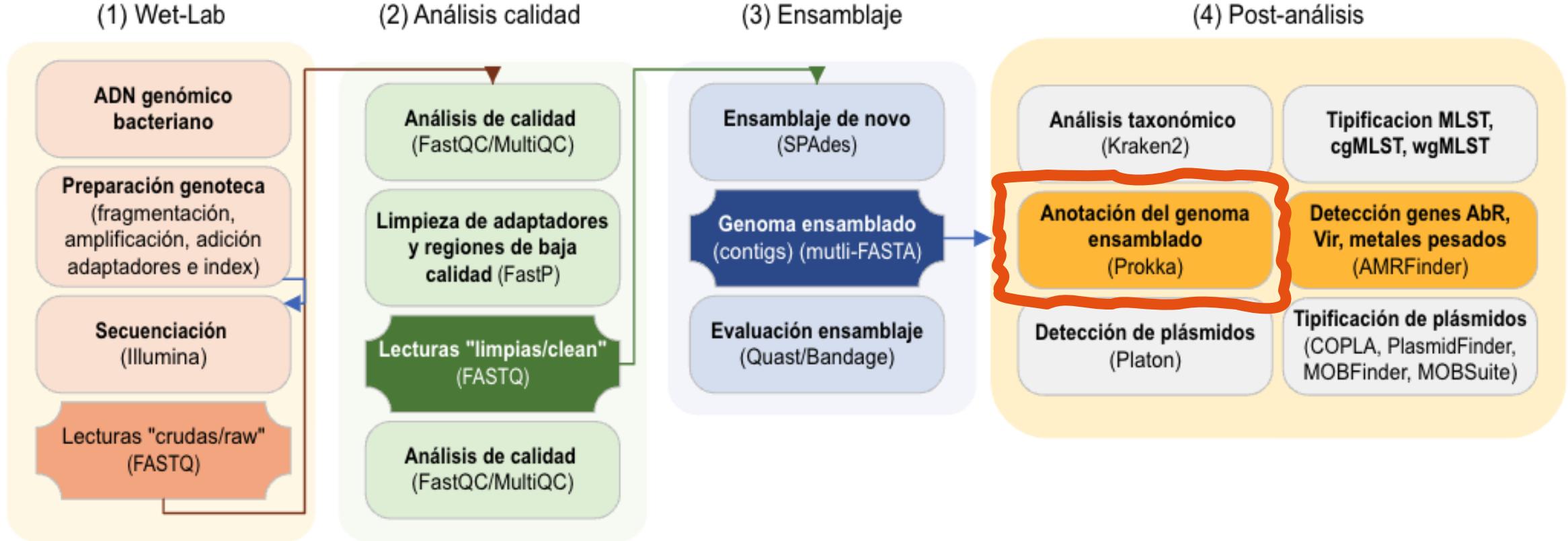
1 941 856 reads

	Worst	Median	Best	<input checked="" type="checkbox"/> Show heatmap
<b>Statistics without reference</b>				
# contigs	1109	1103	1103	
Largest contig	375 037	375 037	375 037	
Total length	6 011 935	6 012 124	6 012 124	
Total length ( $\geq 1000$ bp)	5 601 660	5 601 837	5 601 837	
Total length ( $\geq 10000$ bp)	5 125 214	5 126 462	5 126 462	
Total length ( $\geq 50000$ bp)	4 535 381	4 611 632	4 611 632	
<b>Reads mapping</b>				
Mapped (%)	100.05	100.05	100.05	
Properly paired (%)	99.15	99.16	99.16	
Singlets (%)	0.01	0.01	0.01	
Misjoin mates (%)	0.66	0.66	0.66	
Avg. coverage depth	74	74	74	
Coverage $\geq 1x$ (%)	99.92	99.91	99.91	
<b>Mismatches</b>				
# N's per 100 kbp	0	8.25	8.25	
<a href="#">Extended report</a>				



5.5.

Reconstrucción del genoma:  
anotación del genoma ensamblado



Tras el ensamblaje de lecturas, el siguiente paso es la anotación del genoma. Este es el proceso de **identificar la localización y el papel biológico de los genes encontrados en el ensamblaje**. Los programas utilizados para la anotación de los genomas conllevan el uso de programas externos que nos ayuden en la predicción de las secuencias codificantes de proteínas (CDS), genes de ARN de transferencia, genes de ARN ribosomal y otros, como operones o elementos CRISPR.

El programa de anotación clásico de genomas es **RAST Server**, que ha evolucionado desde una página de anotación web a un protocolo de anotación para múltiples genomas (Aziz et al., 2008; Brettin et al., 2015; Meyer et al., 2008; Overbeek et al., 2014).

Hoy en día los protocolos más utilizados son **NCBI Prokaryotic Genome Annotation Pipeline (PGAP)** (Tatusova et al., 2016) o **Prokka** (Seemann, 2014).

**Siempre debemos tener en cuenta tras el uso de estas herramientas que es necesario un paso de revisión manual para corregir errores potenciales.**

## 5.5.1. Anotación utilizando Prokka

El flujo de trabajo Prokka facilita el proceso de anotación completa de un genoma bacteriano, Archaea o viral.

Prokka está disponible en su última versión en el siguiente enlace:

<https://github.com/tseemann/prokka>

Este flujo de trabajo coordina una serie de herramientas prediseñadas como Blast, predictores de CDS y proteínas como Prodigal, programas de búsqueda de CRISPR, HMMER, etc., para realizar una anotación sencilla de un genoma bacteriano en aproximadamente diez minutos en cualquier ordenador de sobremesa. Este proceso puede complementarse con bases de datos adicionales para refinar la anotación.

**Prokka trabaja con los contigs o scaffolds ensamblados en formato FASTA.** Idealmente, la secuencia debería no tener huecos, pero habitualmente siempre utilizamos un multi-FASTA que procede de nuestro ensamblaje de novo. Este es el único archivo obligatorio.

**El proceso de anotación se realiza basado en herramientas de predicción externas para identificar las coordenadas de los patrones genéticos en los contigs/scaffolds.**

## 5.5.1. Anotación utilizando Prokka

Estas herramientas son:

- Prodigal (Hyatt et al., 2010), para la predicción de secuencias codificantes (CDS).
- RNAmmer (Lagesen et al., 2007), para la búsqueda de genes ribosomales (rARN).
- Aragorn (Laslett y Canback, 2004), para la búsqueda de genes de transferencia de ARN.
- SignalIP (T. N. Petersen et al., 2011), para la búsqueda de péptidos señal.
- Infernal (Nawrocki et al., 2009), para la búsqueda de ARN no codificante (ncARN).

## 5.5.1. Anotación utilizando Prokka

La anotación de los genes codificantes de proteínas se realiza en dos pasos.

Inicialmente, Prodigal identifica las coordenadas de los genes candidatos, pero no describe el producto de ese gen putativo.

La forma tradicional de predecir qué codifica un gen es compararlo con una gran base de datos de secuencias conocidas, generalmente a nivel de secuencia de proteínas, y transferir la anotación de la coincidencia más significativa. Prokka utiliza este método, pero de una manera jerárquica, comenzando por una base de datos confiable más pequeña, pasando a una base de datos de tamaño mediano pero específica del dominio, y finalmente a modelos seleccionados de familias de proteínas.

De manera predeterminada se utilizar un umbral para el e-valor de esta coincidencia de  $10^6$ , incluyendo las siguientes bases de datos:

## 5.5.1. Anotación utilizando Prokka

- Base de datos de proteínas provista por el usuario (opcional). Se espera un archivo multifasta de proteínas anotadas que sean muy fiables. Serán utilizadas como base de datos primaria. Se busca en ellas utilizando BLAST+/Blastp.
- Todas las proteínas bacterianas contenidas en **UniProt**, que tienen evidencia de ser una proteína real o transcripto y que no son un fragmento. Esta base de datos contiene entre 16 000 a 20 000 proteínas y contiene aproximadamente más del 50 % de los genes core de la mayor parte de genomas. A esta se accede vía BLAST+/Blastp.
- Todas las proteínas de genomas bacterianos cerrados en **RefSeq** para un género específico. Si incluimos el género bacteriano entre los comandos de entrada, podrá utilizarse. Las bases de datos varían en tamaño y calidad dependiendo de lo poblado que esté ese género en las bases de datos. Se utiliza BLAST+/Blastp para este análisis.
- Bases de datos basadas en perfiles de **Markov** (hidden Markov model, HMM) incluyendo la base de datos Pfam y TIGRFAMs. Esta búsqueda se realiza utilizando hmmscan del paquete HMMER.

Si no se encuentra ninguna correspondencia en estas bases de datos, la proteína se etiqueta como “proteína hipotética”.

## 5.5.1. Anotación utilizando Prokka

Tras el procesamiento obtendremos los siguientes archivos de salida:

- .fna:** archivo original FASTA que utilizamos como entrada del programa (nucleótidos).
- .faa:** archivo de genes codificantes traducidos en formato FASTA (proteínas).
- .ffn:** archivo de regiones genómicas en formato FASTA (nucleótidos).
- .fsa:** secuencias de los *contigs* preparadas para ser enviadas a las bases de datos como NCBI (nucleótidos).
- .tbl:** tabla con las regiones genómicas y CDS para el envío a las bases de datos.
- .sqn:** archivo en formato SEQUIN editable para su envío a las bases de datos.
- .gbk:** archivo de tipo Genbank que contiene la secuencia y las anotaciones.
- .gff:** archivo de tipo GFF v3 que contiene la secuencia y las anotaciones.
- .log:** archivo de registro del procesamiento de Prokka.
- .txt:** resumen de las estadísticas de la anotación.



## Tema 5 - Ejemplo 4 - Anotación del genoma



En este ejemplo vamos a realizar la anotación de los contigs ensamblados en el paso anterior mediante Prokka. Para ello, vamos a utilizar una base de datos de proteínas de referencia. Aunque existen varias maneras de realizarla, para simplificar el proceso, utilizaremos las proteínas de un genoma de E. coli referencia, como es E.coli K12. Exploraremos las opciones disponibles.

1) Descargamos el archivo de GenBank para E.coli K12, desde el subapartado de [https://www.ncbi.nlm.nih.gov/genome/167?genome\\_assembly\\_id=161521](https://www.ncbi.nlm.nih.gov/genome/167?genome_assembly_id=161521). Nota: recordad bajar el archivo GenBank (Full).

2) Extraemos las proteínas desde este archivo, con la función de Prokka siguiente:

```
prokka-genbank_to_fasta_db GCF_000005845.2_ASM584v2_genomic.gbff > ref_prots.faa
```

3) Ejecutamos prokka con los siguientes parámetros:

```
prokka --outdir prokka_Ecoli --addgenes --addmrna --genus Escherichia --species coli --kingdom Bacteria --usegenus --proteins ref_prots.faa --mincontiglen 0 contigs.fasta
```

Recordad que para más opciones debéis consultar el manual de Prokka o la web <https://github.com/tseemann/prokka>.

Resultados ejercicio: [Tema5\\_Ejemplo4\\_Prokka](#)

Tiempo aprox. 5 min



Genome

Genome ▾

[Search](#)

Limits Advanced

Help

#### Escherichia coli str. K-12 substr. MG1655

Download sequences in FASTA format for [genome](#), [protein](#)

Download genome annotation in [GFF](#), [GenBank](#) or [tabular](#) format

BLAST against Escherichia coli [genome](#), [protein](#)

#### All 32626 genomes for species:

Browse the [list](#)

Download sequence and annotation from [RefSeq](#) or [GenBank](#)

**NEW** Try the NCBI Datasets [Taxonomy page](#) - a new way to access genomic data, including reference genomes

Display Settings: ▾ [Overview](#)

Send to: ▾

[Organism Overview](#) : [Genome Assembly and Annotation report](#) : [Genome Neighbor report](#)

## Escherichia coli str. K-12 substr. MG1655

Model organism for genetics, physiology, biochemistry

Lineage: [Bacteria](#)[34407]; [Proteobacteria](#)[11441]; [Gammaproteobacteria](#)[4231]; [Enterobacterales](#)[735]; [Enterobacteriaceae](#)[390]; [Escherichia](#)[7]; [Escherichia coli](#)[1]; [Escherichia coli](#) K-12[1]; [Escherichia coli str. K-12 substr. MG1655](#)[1]

### Summary

**Submitter:** Univ. Wisconsin  
**Assembly level:** Complete Genome  
**Morphology:** Gram:Negative, Shape:Bacilli, Motility:Yes  
**Environment:** OxygenReq:Facultative, OptimumTemperature:37, TemperatureRange:Mesophilic, Habitat:HostAssociated  
**Assembly:** GCA\_000005845.2 ASM584v2 **scaffolds:** 1 **contigs:** 1 **N50:** 4,641,652 **L50:** 1  
**BioProjects:** PRJNA57779, PRJNA225  
**Statistics:** total length (Mb): 4.641652  
protein count: 4298  
GC%: 50.8

### Genome Neighbors

**Closest species reference genome:** [Escherichia coli O157:H7 str. Sakai](#) Symmetrical identity: 80.0895%

**Closest genome:** [Escherichia coli](#) Symmetrical identity: 100%

**Genome Group:** 1 genomes at symmetrical identity 100% (See [Genome Neighbor report](#))

### Related information

Assembly

BioProject

Gene

Components

Protein

PubMed

Taxonomy

### Recent activity

[Turn Off](#) [Clear](#)

[Escherichia coli str. K-12 substr. MG1655](#)

Genome

[See more...](#)



```
[20:38:35] prokka_Ecoli/PROKKA_12182022.fna
[20:38:35] prokka_Ecoli/PROKKA_12182022.ffn
[20:38:35] prokka_Ecoli/PROKKA_12182022.tsv
[20:38:35] prokka_Ecoli/PROKKA_12182022.err
[20:38:35] prokka_Ecoli/PROKKA_12182022.faa
[20:38:35] prokka_Ecoli/PROKKA_12182022.tbl
[20:38:35] prokka_Ecoli/PROKKA_12182022.gff
[20:38:35] prokka_Ecoli/PROKKA_12182022.fsa
[20:38:35] prokka_Ecoli/PROKKA_12182022.txt
[20:38:35] prokka_Ecoli/PROKKA_12182022.sqn
[20:38:35] prokka_Ecoli/PROKKA_12182022.log
[20:38:35] prokka_Ecoli/PROKKA_12182022.gbk
[20:38:35] Annotation finished successfully.
[20:38:35] Walltime used: 4.97 minutes
[20:38:35] If you use this result please cite the Prokka paper:
[20:38:35] Seemann T (2014) Prokka: rapid prokaryotic genome annotation. Bioinformatics. 30(14):2068-9.
[20:38:35] Type 'prokka --citation' for more details.
[20:38:35] Thank you, come again.
```

```
organism: Escherichia coli strain
contigs: 1109
bases: 6011935
CDS: 5601
gene: 5712
mRNA: 5712
rRNA: 9
repeat_region: 1
tRNA: 101
tmRNA: 1
PROKKA_12182022.txt (END)
```

## 5.5.2. Visualización de las anotaciones

Una vez anotado el genoma de manera automática, es importante realizar un proceso de curado manual de la anotación. Este proceso es largo y tedioso, por lo que el paso de anotación con bases de datos específicas puede facilitar sustancialmente los pasos posteriores.

Se pueden visualizar los archivos de anotación en programas como [Artemis](#) (Carver et al., 2012) o [CLC Sequence Viewer/CLC Genomics Workbench](#) (Latest improvements for CLC Sequence Viewer - Current line - Qiagen Bioinformatics, n.d.), donde se pueden modificar las regiones detectadas. Existen programas más avanzados, pero que requieren una suscripción de pago. Los archivos habitualmente utilizados para revisar las anotaciones son los archivos [GenBank \(GBK\)](#) o [GFF/GTF](#). Ambos tipos de archivos son proporcionados por Prokka.

**Artemis:** Aunque es una herramienta bastante antigua (Berriman y Rutherford, 2003; Carver et al., 2012; Rutherford et al., 2000) es una de las pocas herramientas de visualización que nos permite editar los archivos de manera gratuita sin pagar licencias. Además de la lectura de archivo GFF, se pueden leer archivos de tipo GenBank (siempre que contengan la secuencia en su interior) o FASTA. Se pueden añadir anotaciones a estos archivos en diferentes capas de información, incluso realizar una predicción de genes desde el propio programa, aunque no es una opción muy depurada.

Para aprender el funcionamiento de Artemis en detalle, se recomienda la lectura del manual *Viewing and annotating sequence data with Artemis* de Berriman y Rutherford (2003), así como la visualización del videotutorial:

<https://www.youtube.com/watch?v=3m84K7I7Omk>

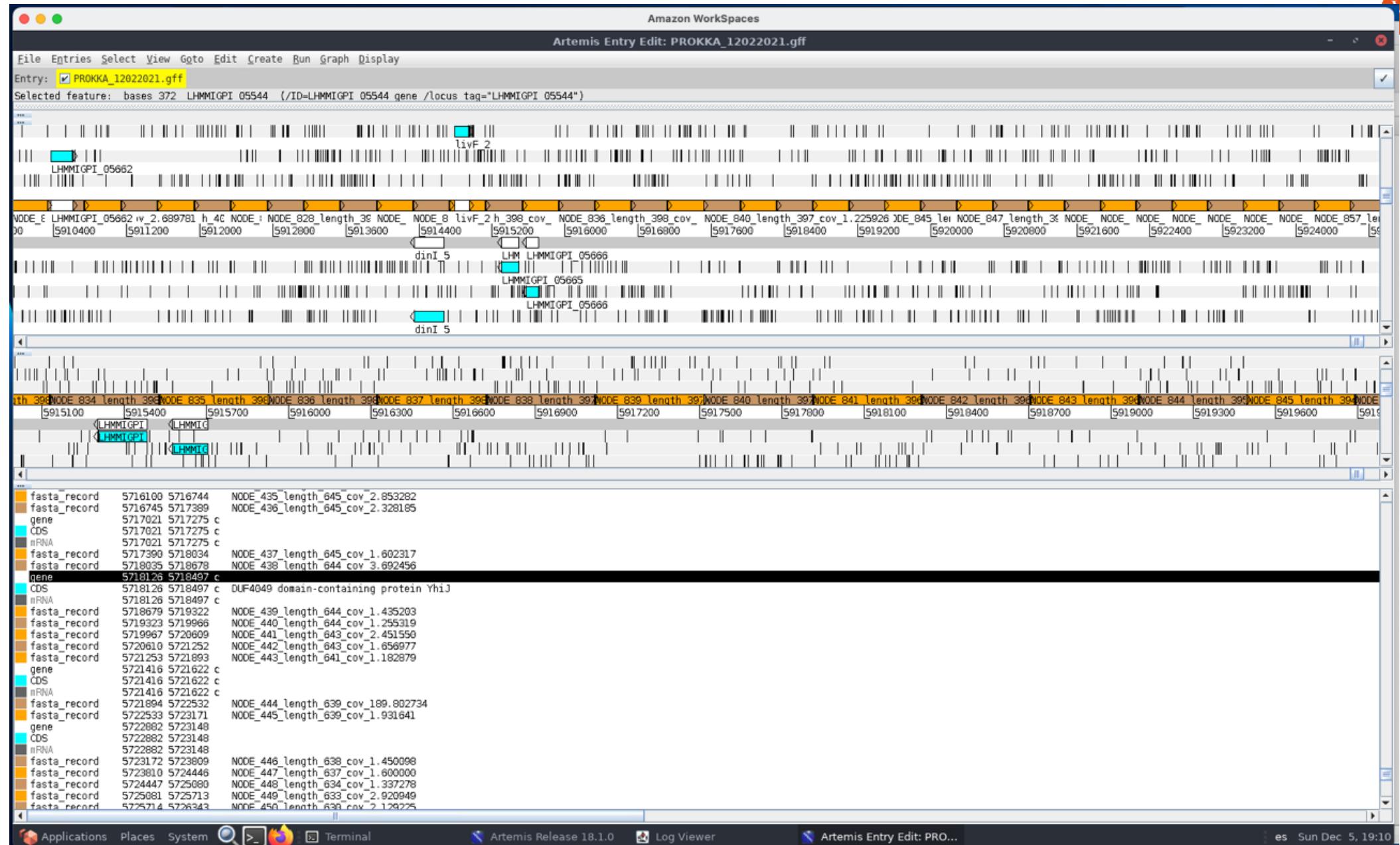


## Tema 5 - Ejemplo 5 - Visualización de las anotaciones



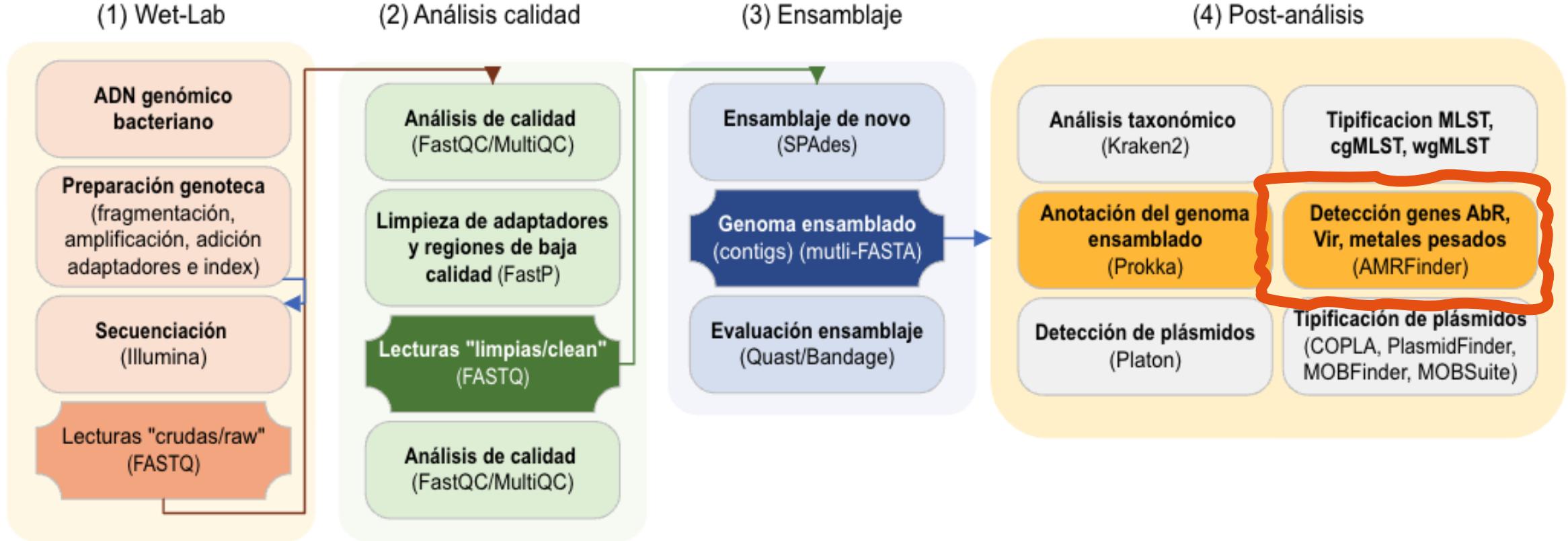
En este ejemplo vamos a visualizar las anotaciones del genoma de *E. coli* del ejemplo anterior, para analizar las regiones detectadas y anotadas. La apertura del programa se realiza desde la terminal, simplemente indicando “art”. Para una mayor sencillez podemos abrir directamente el archivo GFF indicando “art file.gff”.

Nota: el archivo GFF procede de la anotación del ejemplo 4 de este tema. Si no tenéis el archivo, lo podéis encontrar [I Tema5\\_Ejemplo5\\_ArtemisCLC](#)



## 5.7.

Detección y anotación de regiones de interés: genes de resistencia a antibióticos y virulencia



Como hemos visto hasta este momento, la secuenciación del genoma completo de los organismos tiene grandes utilidades.

En esta sección, veremos que la predicción de los genes de resistencia a antibióticos o virulencia a partir de la secuencia del genoma completo es una herramienta muy útil en la predicción de fenotipos, monitorización de la dispersión de este tipo de genes y en epidemiología genómica.

Para esta finalidad, existen diferentes procedimientos:

- **Mapear las lecturas secuenciadas directamente sobre una base de datos de referencia**, que contenga las secuencias que queremos detectar. Esta aproximación se utiliza habitualmente con datos de metagenómica, puesto que no es necesario ensamblar el genoma completo para poder realizar esta predicción. El macheo puede realizarse directamente con las lecturas completas, como la que ejecutan los programas SRST2 (Inouye et al., 2014) o GROOT (Rowe y Winn, 2018). Por otra parte, otra aproximación más rápida aún es machear k-meros de las lecturas secuenciadas frente a la base de datos de referencia, siendo un método rápido y evitando los pasos de ensamblaje y anotación, pero incrementando el riesgo de falsos positivos debido a errores de secuenciación o contaminación de ADN. Esta última aproximación la utilizan programas como PhenotypeSeeker (Aun et al., 2018) o KmerResistance (Clausen et al., 2016). Ambas aproximaciones son muy útiles cuando tenemos un gran número de genomas y queremos seleccionar aquellos que portan alguna característica especial para su procesamiento posterior.

- **Mapear los contigs/scaffolds ensamblados previamente utilizando el algoritmo Basic Local Alignment Search Tool (BLAST)** para encontrar todos los posibles genes de interés entre esta secuencia y la base de datos. Los contigs/scaffolds pueden compararse frente a bases de datos públicas disponibles o bien frente a bases de datos diseñadas a medida. Este **procedimiento es más lento**, puesto que requiere un paso previo de ensamblaje e, incluso, de anotación de secuencias codificantes. Esto puede ser **costoso en tiempo y en recursos computacionales**. Asimismo, **este método solo es fiable para genomas sencillos**, mientras que ve limitada su aplicación en poblaciones de genomas complejos.

Independientemente del método utilizado, es crucial utilizar una **base de datos fiable y curada** para obtener resultados fidedignos sobre el fenotipo de resistencia y virulencia. Entre estas bases de datos encontramos:

- **ResFinder** (E. Zankari et al., 2012) (para genes de resistencia a antibióticos adquiridos), **PointFinder** (Ea Zankari et al., 2017) (para mutaciones puntuales) y **VirulenceFinder** (Joensen et al., 2014) (para factores de virulencia). Todas ellas han sido implementadas vía web, pero también es posible descargar las secuencias y utilizarlas por vía de comandos:  
<https://bitbucket.org/genomicepidemiology/workspace/projects/DB>
- **Comprehensive Antibiotic Resistance Database (CARD)** (Jia et al., 2017) y **ARG-ANNOT** (Gupta et al., 2014) son dos de las bases de datos más utilizadas para genes de resistencia a antibióticos; mientras que **Virulence Factor Data Base (VFDB)** (Liu et al., 2019; VFDB: Virulence Factors Database, n.d.) lo es para genes de virulencia.
- La base de datos **AMRFinder**, perteneciente a NCBI, es una herramienta para detectar genes de resistencia a antibióticos utilizando la anotación de proteínas o secuencias nucleotídicas obtenidas directamente de NCBI y una colección curada de modelos de Markov:  
<https://www.ncbi.nlm.nih.gov/pathogens/antimicrobial-resistance/AMRFinder/>

Aunque en algunos casos estas bases de datos disponen de su propio programa de búsqueda, en todos los casos basados en BLAST, en todo caso podemos descargar estas bases de datos, manipularlas si fuese necesario y utilizar BLAST de manera local. Obtendremos una tabla manipulable y fácil de interpretar. Cabe destacar que aquellos genes de interés que no están presentes en la base de datos utilizada, obviamente, no podrán ser detectados.



## Tema 5 - Ejemplo 6 - Análisis AbR y Vir



En este ejemplo vamos a utilizar la anotación Prokka procedente del Ejemplo 4, donde tenemos los archivos de CDS y proteínas calculados (descarga si no los tienes [Tema5\\_Ejemplo4\\_Prokka](#)). Utilizaremos **AMRFinder** para la detección de genes de resistencia a antibióticos, virulencia y resistencia a metales pesados.

Este programa está instalado en el environment conda de trabajo de este tema. Toda la información de instalación y uso está en la web <https://www.ncbi.nlm.nih.gov/pathogens/antimicrobial-resistance/AMRFinder/>.

1) Actualizar la base de datos a su última versión:

```
amrfinder --update
```

2) Seleccionar el organismo a utilizar preferentemente, buscando entre los disponibles como:

```
amrfinder --list_organisms
```

3) Ejecutamos el programa, atendiendo a los parámetros de cobertura e identidad utilizados

```
amrfinder --organism Escherichia -n PROKKA_12022021.ffn -c 0.8 -i 0.9 --plus --log amrfinder.log > AMRFinder_out.csv
```

4) Analizemos:

- a. ¿Qué tipos de genes se detectan con esta base de datos?
- b. ¿Qué genes de resistencia encuentras? ¿y de virulencia?
- c. ¿existen genes de resistencia a metales pesados?

Soluciones a este ejercicio [Tema5\\_Ejemplo6\\_AbR\\_Vir](#)

Tiempo aprox: 2 – 5 min

```
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra prokka_Ecoli]$ amrfinder --update
Running: amrfinder --update
Software directory: '/home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/bin/'
Software version: 3.11.2
The number of threads cannot be greater than 2 on this computer
The current number of threads is 4, reducing to 2
Running: /home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/bin/amrfinder_update -d /home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/share/amrfinderplus/data
Looking up the published databases at https://ftp.ncbi.nlm.nih.gov/pathogen/Antimicrobial_resistance/AMRFinderPlus/database/

WARNING: Cannot get the latest published database version for the current software minor version 3.11.
The latest published database version 2022-10-11.2 for the latest published software minor version 3.10 will be used instead.

Downloading AMRFinder database version 2022-10-11.2 into '/home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/share/amrfinderplus/data/2022-10-11.2/'
Indexing
Database directory: '/home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/share/amrfinderplus/data/2022-10-11.2'
Database version: 2022-10-11.2
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra prokka_Ecoli]$ █
```

```
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra prokka_Ecoli]$ amrfinder --list_organisms
Running: amrfinder --list_organisms
Software directory: '/home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/bin/'
Software version: 3.11.2
The number of threads cannot be greater than 2 on this computer
The current number of threads is 4, reducing to 2
Database directory: '/home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/share/amrfinderplus/data/2022-10-11.2'
Database version: 2022-10-11.2

Available --organism options: Acinetobacter_baumannii, Burkholderia_cepacia, Burkholderia_pseudomallei, Campylobacter, Clostridioides_difficile, Enterococcus_faecalis, Enterococcus_faecium, Escherichia_klebsiella, Neisseria, Pseudomonas_aeruginosa, Salmonella, Staphylococcus_aureus, Staphylococcus_pseudintermedius, Streptococcus_agalactiae, Streptococcus_pneumoniae, Streptococcus_pyogenes, Vibrio_cholerae
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra prokka_Ecoli]$
```

Protein identifier	Contig id	Start	End	Strand	Gene symbol	Sequence name	Scop	Element type	Element subtyp	Class	Subclass	Method	Targ		
NA	OMPBAEK_00054	1	1230+	+	mdtM	multidrug efflux MFS transporter MdtM	plus	AMR	AMR	EFFFLUX	EFFFLUX	BLASTX			
NA	OMPBAEK_02321	34	1164+	+	blaEC	BlaEC family class C beta-lactamase	plus	AMR	AMR	BETA-LACTAM	BETA-LACTAM	BLASTX			
NA	OMPBAEK_02362	10	1098+	+	pmrB	Escherichia coliSTIN resistant PmrB	core	AMR	POINT	COLISTIN	COLISTIN	POINTX			
NA	OMPBAEK_02918	1	1356+	+	gipT_E448K	Escherichia fosfomycin resistant GipT	core	AMR	POINT	FOSFOMYCIN	FOSFOMYCIN	POINTX			
NA	OMPBAEK_04335	1	576+	+	terD	tellurium resistance membrane protein TerD	plus	STRESS	METAL	TELLURIUM	TELLURIUM	BLASTX			
NA	OMPBAEK_04336	1	1038+	+	terC	tellurium resistance membrane protein TerC	plus	STRESS	METAL	TELLURIUM	TELLURIUM	BLASTX			
NA	OMPBAEK_04337	1	453+	+	terB	tellurium resistance membrane protein TerB	plus	STRESS	METAL	TELLURIUM	TELLURIUM	BLASTX			
NA	OMPBAEK_04339	1	579+	+	terZ	tellurium resistance-associated protein TerZ	plus	STRESS	METAL	TELLURIUM	TELLURIUM	BLASTX			
NA	OMPBAEK_04346	1	465+	+	terW	tellurium resistance protein TerW	plus	STRESS	METAL	TELLURIUM	TELLURIUM	BLASTX			
NA	OMPBAEK_05285	1	330+	+	emrE	multidrug efflux SMR transporter EmrE	plus	STRESS	BIOCIDE	EFFFLUX	EFFFLUX	BLASTX			
NA	OMPBAEK_00140	1	1419+	+	espX1	type III secretion system effector EspX1	plus	VIRULENCE	VIRULENCE	NA	NA	BLASTX			
NA	OMPBAEK_00910	1	522+	+	lpfA1	long polar fimbria major subunit LpfA1	plus	VIRULENCE	VIRULENCE	NA	NA	BLASTX			
NA	OMPBAEK_01510	1	4242+	+	fdeC	intimin-like adhesin FdeC	plus	VIRULENCE	VIRULENCE	NA	NA	BLASTX			
NA	OMPBAEK_01995	1	978+	+	nleB2	type III secretion system effector arginine glycosyltransferase NleB2	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_01996	28	1017+	+	nleC	type III secretion system effector zinc metalloprotease NleC	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_02928	1	936+	+	espB	type III secretion system LEE translocon pore-forming subunit EspB	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_02930	1	576+	+	espA	type III secretion system LEE translocon filament protein EspA	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_02933	1	2802+	+	eae	intimin type gamma	plus	VIRULENCE	VIRULENCE	INTIMIN	GAMMA	BLASTX			
NA	OMPBAEK_02935	1	1674+	+	tir	type III secretion system LEE translocated intimin receptor Tir	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_03038	1	600+	+	lpfA2	long polar fimbria major subunit LpfA2	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_04211	1	651+	+	espJ	type III secretion system effector ADP-ribosyltransferase EspJ	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_04332	1	2088+	+	iha	bifunctional siderophore receptor/adhesin Iha	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_04622	1	2994+	+	ehxA	enterohemolysin EhxA	plus	VIRULENCE	VIRULENCE	NA	NA	BLASTX			
NA	OMPBAEK_04637	1	1965+	+	etpD	variant type II secretion system secretin EtpD	plus	VIRULENCE	VIRULENCE	NA	NA	BLASTX			
NA	OMPBAEK_04672	1	1359+	+	espK	type III secretion system effector EspK	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_04886	1	987+	+	nleB	type III secretion system effector arginine glycosyltransferase NleB	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_04924	1	9507+	+	toxB	toxin B	plus	VIRULENCE	VIRULENCE	NA	NA	BLASTX			
NA	OMPBAEK_04990	1	1323+	+	nleA	type III secretion system effector NleA	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_04995	1	291+	+	iss	increased serum survival lipoprotein Iss	plus	VIRULENCE	VIRULENCE	NA	NA	PARTIAL_CONTIG_ENDX			
NA	OMPBAEK_05011	1	957+	+	stxA2c	Shiga toxin Stx2c subunit A	plus	VIRULENCE	VIRULENCE	STX2	STX2C	EXACTX			
NA	OMPBAEK_05012	1	267+	+	stxB2a	Shiga toxin Stx2a subunit B	plus	VIRULENCE	VIRULENCE	STX2	STX2A	EXACTX			
NA	OMPBAEK_05050	1	291+	+	iss	increased serum survival lipoprotein Iss	plus	VIRULENCE	VIRULENCE	NA	NA	PARTIAL_CONTIG_ENDX			
NA	OMPBAEK_05087	1	3900+	+	espP	serine protease autotransporter EspP	plus	VIRULENCE	VIRULENCE	NA	NA	EXACTX			
NA	OMPBAEK_05096	1	2208+	+	katP	catalase/peroxidase KatP	Subclass	Method	Target leng	Reference sequence leng	% Coverage of reference sequen	% Identity to reference sequen	Alignment leng	Accession of closest sequen	Name of closest sequence
NA	OMPBAEK_05262	1	423+	+	subB	subtilase ABS cytolysin subunit B	EFFFLUX	BLASTX	410	410	100	98.05	410AAC7293.1	multidrug efflux MFS transporter MdtM	
NA	OMPBAEK_05263	1	1041+	+	subA	subtilase ABS cytolysin subunit A	BETA-LACTAM	BLASTX	377	377	100	98.14	377WP_063610930.1	extended-spectrum class C beta-lactamase EC-1	
NA	OMPBAEK_05530	1	525+	+	lpfA	long polar fimbria major subunit LpfA-O113	COLISTIN	POINTX	363	363	100	99.17	363WP_001300761.1	two-component system sens or histidine kinase P	
						FOSFOMYCIN	POINTX	452	452	100	99.12	452WP_000948731.1	glycerol-3-phosphate transporter GipT		
						TELLURIUM	BLASTX	192	192	100	95.31	192AAAG8929.1	tellurium resistance membrane protein TerD		
						TELLURIUM	BLASTX	346	346	100	90.46	346AAA9291.1	tellurium resistance membrane protein TerC		
						TELLURIUM	BLASTX	151	151	100	90.73	151ACI12150.1	tellurium resistance membrane protein TerB		
						TELLURIUM	BLASTX	193	193	100	98.96	193AAAG6846.1	tellurium resistance-associated protein TerZ		
						TELLURIUM	BLASTX	155	155	100	99.35	155AAC4736.1	tellurium resistance protein TerW		
						EFFFLUX	BLASTX	110	110	100	98.18	110CAA77936.1	multidrug efflux SMR transporter EmrE		
						NA	BLASTX	473	473	100	99.79	473ADD54679.1	type III secretion system effector EspX1		
						NA	BLASTX	174	174	100	100	174OPH66324.1	long polar fimbria major subunit LpfA1		
						NA	BLASTX	1414	1416	99.93	92.23	1415ABE05822.1	intimin-like adhesin FdeC		
						NA	EXACTX	326	326	100	100	326CAS0859.1	type III secretion system effector arginine glycosyltransferase		
						NA	EXACTX	330	330	100	100	330AACG5141.1	type III secretion system effector zinc metalloprotease		
						NA	EXACTX	312	312	100	100	312AAC58818.1	type III secretion system LEE translocon pore-forming subunit		
						NA	EXACTX	192	192	100	100	192CAA73506.1	type III secretion system LEE translocon filament		
						NA	EXACTX	934	934	100	99.79	934ACM67148.1	intimin type gamma		
						NA	EXACTX	558	558	100	100	558AAC58825.1	type III secretion system LEE translocated intimin		
						NA	EXACTX	200	200	100	100	200BAB38093.1	long polar fimbria major subunit LpfA2		
						NA	EXACTX	217	217	100	100	217AAG56990.1	type III secretion system effector ADP-ribosyltran		
						NA	EXACTX	696	696	100	100	696AAC523.21	bifunctional siderophore receptor/adhesin Iha		
						NA	EXACTX	998	998	100	99.9	998BAA31774.1	enterohemolysin EhxA		
						NA	BLASTX	656	656	99.85	94.5	655BAI33940.1	variant type II secretion system secretin EtpD		
						NA	EXACTX	453	453	100	100	453BAB34991.1	type III secretion system effector EspK		
						NA	EXACTX	329	329	100	100	329BAB37280.1	type III secretion system effector arginine glycosyltransferase		
						NA	BLASTX	3169	3169	100	99.94	3169BAA31815.1	toxin B		
						NA	EXACTX	441	441	100	100	441AAK16940.1	type III secretion system effector NleA		
						NA	PARTIAL_CONTIG_ENDX	97	113	85.84	91.75	97AAN80033.1	increased serum survival lipoprotein Iss		
						NA	EXACTX	319	319	100	100	319CAA31814.1	Shiga toxin Stx2c subunit A		
						NA	EXACTX	89	89	100	100	89CAA30715.1	Shiga toxin Stx2a subunit B		
						NA	PARTIAL_CONTIG_ENDX	97	113	85.84	93.81	97AAN80033.1	increased serum survival lipoprotein Iss		
						NA	BLASTX	1300	1300	100	100	1300BAA31836.1	serine protease autotransporter EspP		
						NA	BLASTX	736	736	100	100	736BAA31832.1	catalase/peroxidase KatP		
						NA	BLASTX	141	141	100	99.29	141AAT68784.1	subtilase ABS cytolysin subunit B		
						NA	BLASTX	347	347	100	100	347AAZ76524.1	subtilase ABS cytolysin subunit A		
						NA	BLASTX	175	190	92.11	99.43	175AAT76975.1	long polar fimbria major subunit LpfA-O113		

<https://github.com/ncbi/amr/wiki/Interpreting-results>

## The "Scope" column and the `--plus` option

AMRFinderPlus splits the database into two subsets. By default it only returns genes in the 'core' subset.

- "Core": this subset includes highly curated, AMR-specific genes and proteins from the Bacterial Antimicrobial Resistance Reference Gene Database (BioProject PRJNA313047), plus point mutations. The sources of input for this curated database include: 1) allele assignments, 2) exchanges with other external curated resources, 3) reports of novel antimicrobial resistance proteins in the literature.
- "Plus": this subset includes genes related to biocide and stress resistance, general efflux, virulence, or antigenicity, or AMR genes whose presence/absence are unlikely to affect phenotype and/or is highly uncertain. These genes are only shown if the `--plus` option is used.

## The "Method" column

Important information about how likely the hit is to be functional and/or how novel a hit is can be found in the "method" column. A description of possible values is listed on the [Running AMRFinderPlus page](#). Note that for BLAST-based "Methods" there is a suffix of either 'X' or 'P' that indicates whether it was found using translated genomic sequence or protein sequence.

### X, P, or N suffix

The methods that can apply to either nucleotide or protein sequence have a suffix appended to designate which type of search was used for that line in the report. The suffix indicates whether the line was identified in a protein (P) or nucleotide translated (X) input file. Nucleotide BLAST (BLASTN) is used to identify nucleotide point mutations and POINTN is used as the Method for those hits.

Note as of version 3.10.11 BLASTX / POINTX / EXACTX results may actually come from `tblastn`. Tblastn is used instead of blastx to increase speed in some cases.

### EXACTX, EXACTP, ALLELEX, and ALLELEP

These mean that we have an amino-acid sequence identical protein in the AMRFinderPlus database. The ALLELE method is reserved for genes that have an allele numbering scheme, so the "Gene symbol" for an ALLELE hit is really an allele symbol specific for that exact sequence. EXACT matches have an identical amino-acid sequence in the database, but the family does not have alleles so the same gene symbol may be shared with other closely related sequences. Note that if you run a combined search (including `-p`, `-n`, and `-g` options) and you get an ALLELEX or EXACTX result that means that the annotation did not contain the protein.

### BLASTX and BLASTP

"BLAST" hits are hits that are < 100% identical to a database protein, but at coverage > 90%. The percent identity cutoff is, by default, 90%, but may be higher or lower if it was manually curated. Manual curation of BLAST parameters usually occurs for 'plus' genes where no HMM and cutoff have been curated.

### PARTIALX, PARTIALP, PARTIAL\_CONTIG\_ENDX, and PARTIAL\_CONTIG\_ENDP

Hits < 90% of the length of the database protein are called either "PARTIAL" if the hit is internal to a contig or "PARTIAL\_CONTIG\_END" if the gene could have been broken by a contig boundary. "PARTIAL\_CONTIG\_END" require that the protein alignment ends within two nucleotides of the end of the contig with an orientation such that the end of the protein would extend off the end of the contig. Because assemblers sometimes split genes over multiple contigs, genes that are PARTIAL\_CONTIG\_END are often full-length in reality. Note that searches using only protein FASTA files (`-p <protein_fasta>`) do use contig location and so all < 90% coverage hits will just be called PARTIAL.

Note that by default AMRFinderPlus does not identify gene fragments that cover < 50% of the reference, so alignments from genes marked PARTIAL cover between 50% and 90% of the reference. The default minimum coverage can be changed by using the `--coverage_min` option (see [Usage \(syntax/options\)](#) for more information).

### INTERNAL\_STOP

These are genes that when translated from genomic sequence have a stop codon before the end of the database protein. These are less likely to be functional, and can only be assessed if the `-n <nucleotide_fasta>` option is used.

### HMM

HMM-only hits happen when the criteria for a BLAST hit is not met and an HMM match is above the curated cutoff for an HMM that has been created for that gene or gene family. These will usually be distant relatives of known gene families and may be candidates for a new gene family. Occasionally, partial proteins that have diverged enough from known database proteins to not meet the BLAST cutoffs will show up as HMM-only hits.

### POINTX, POINTP, and POINTN

Point mutation detection is only enabled when an `--organism` option is included for a particular taxonomic group. AMRFinderPlus does not report all point mutations in those genes, only ones that have been found in papers describing their functional relevance. Thus the absence of a POINTP, POINTX, or POINTN result does not mean that there are no functional point mutations, only that AMRFinderPlus was unable to find a gene with known point mutations. This could be either because the gene does not exist in the assembly, or the functional point mutation was not one included in the database. See [The Pathogen Detection Reference Gene Browser](#) for a list of known point mutations. The `--mutation_all` option will create a file with all genotype calls made by AMRFinderPlus. With that output you can assess whether or not a specific gene was identified in the query sequence.

## Element Type and Subtype

These fields describe the classification of the AMRFinderPlus gene. "Element type" is split into 3 categories, AMR, STRESS, or VIRULENCE. "Element subtype" is a duplicate of "Element type" unless a more specific category has been defined.

Element type	Element subtype	Description
AMR	AMR	Antimicrobial resistance gene
AMR	AMR-SUSCEPTIBLE	Not used in AMRFinderPlus output, but in the Reference Gene Catalog (see note below)
AMR	POINT	Known point mutation associated with antimicrobial resistance
VIRULENCE	VIRULENCE	Virulence gene
VIRULENCE	ANTIGEN	Gene codes for a known antigen (these are often used for typing)
STRESS	ACID	Acid resistance gene
STRESS	BIOCIDE	Biocide resistance gene
STRESS	HEAT	Heat resistance gene
STRESS	METAL	Metal resistance gene

### A note about subtype AMR-SUSCEPTIBLE and *Streptococcus pneumoniae*

This is used for Penicillin-resistant *Streptococcus pneumoniae* pbp alleles, which are defined as those alleles which are below a protein identity threshold to a set of pbp1a, pbp2b, or pbp2x reference alleles: the thresholds are 99%, 99%, and 98% for pbp1a, pbp2b, and pbp2x respectively. The susceptible reference alleles can be found in NCBI's [Reference Gene Catalog](#) where they have subtype AMR-SUSCEPTIBLE to indicate that the reference is the susceptible version. These alleles may also confer resistance to other beta-lactam antibiotics. If you are using AMRFinderPlus with *Streptococcus pneumoniae* pbp genes you may need to analyze further as necessary. Subtype AMR-SUSCEPTIBLE will not appear in AMRFinderPlus output because only putatively resistant forms will be identified.

## Class and Subclass

These are classifications of the effect of the gene on resistance to various stresses. Be aware that genotype and phenotype may not correspond (see also [note regarding genotype and phenotype](#)). The class and subclass fields are also used to add typing information in the cases of stx and intimins. These fields are blank for many "plus" genes.

For a list of all possible combinations of class and subclass see [class-subclass](#). For AMR genes "Class" can be thought of as drug class, and 'Subclass' contains a more specific drug designations where known. While most subclass designations are self-explanatory, a few others have particular meanings. Specifically, "CEPHALOSPORIN" is equivalent to the Lahey 2be definition; "CARBAPENEM" means the protein has carbapenemase activity, but it might or might not confer resistance to other beta-lactams; "QUARTERNARY AMMONIUM" are quarternary ammonium compounds. Where the phenotypic information is incomplete, contradictory, or unclear, the "Class" value is used for the "Subclass" value.

For some antigen and virulence genes that are often referred to by type, specific type information is included here as follows:

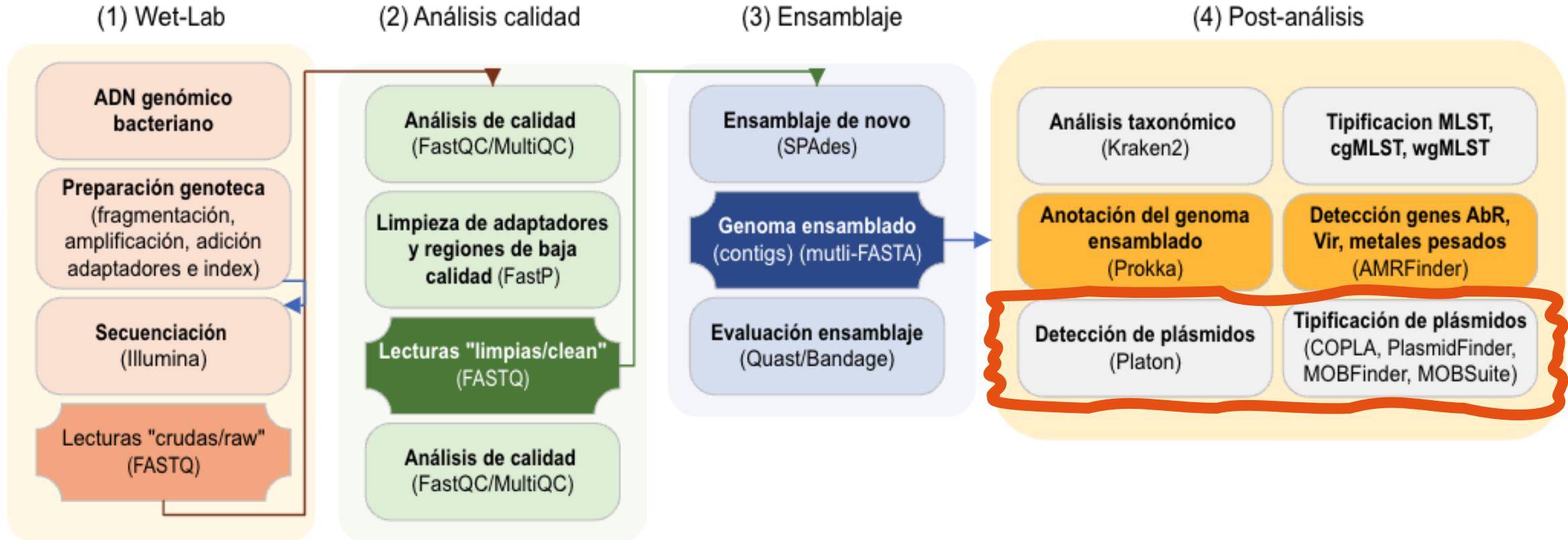
For eae (Intimin) genes the "Class" is INTIMIN, while the subclass contains the intimin type (ALPHA, BETA, EPSILON, GAMMA, IOTA, LAMBDA, or RHO).

For stx (Siga toxin) genes the possibilities for "Class" and "Subclass" provide typing information as follows:

Class	Subclass
STX1	STX1A
STX1	STX1B
STX1	STX1C
STX1	STX1D
STX2	STX2A
STX2	STX2B
STX2	STX2C
STX2	STX2D
STX2	STX2E
STX2	STX2F
STX2	STX2G

5.7.

Reconstrucción del genoma.  
Elementos Genéticos Móviles  
(EGMs). Plásmidos



Los elementos genéticos móviles (EGM) son todos aquellos elementos genéticos que juegan un papel importante en la transferencia horizontal de material genético entre bacterias.

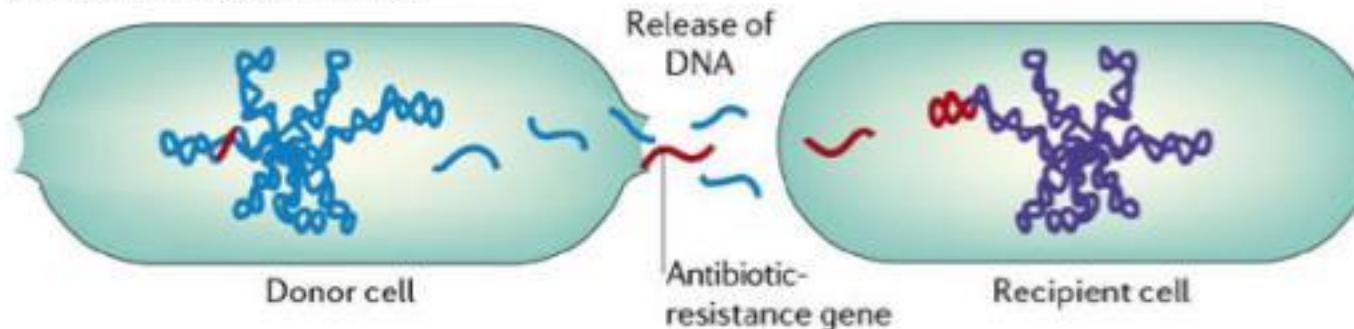
Los organismos procariotas intercambian ADN de manera horizontal mediante tres procesos, donde la transducción y la conjugación dependen de EGM especializados, donde se incluyen los plásmidos y algunos bacteriófagos:

- Transformación, donde el paso del material genético está mediado por la captación de ADN libre.
- Conjugación, donde la transferencia del material genético está mediada por plásmidos o elementos integrativos conjugativos (ICE). El contacto entre células, a través de un pilus de conjugación es necesario para esta transferencia.
- Transducción, mediada por la presencia de un bacteriófago que encapsula el ADN entre su material genético y lo libera a una célula nueva.

Los organismos procariotas poseen una tercera clase de EGM que son los transposones, capaces de moverse y reorganizar el ADN en la propia célula. Los transposones se mueven entre distintas células a través de los plásmidos, fagos o los elementos integrativos conjugativos (ICE).

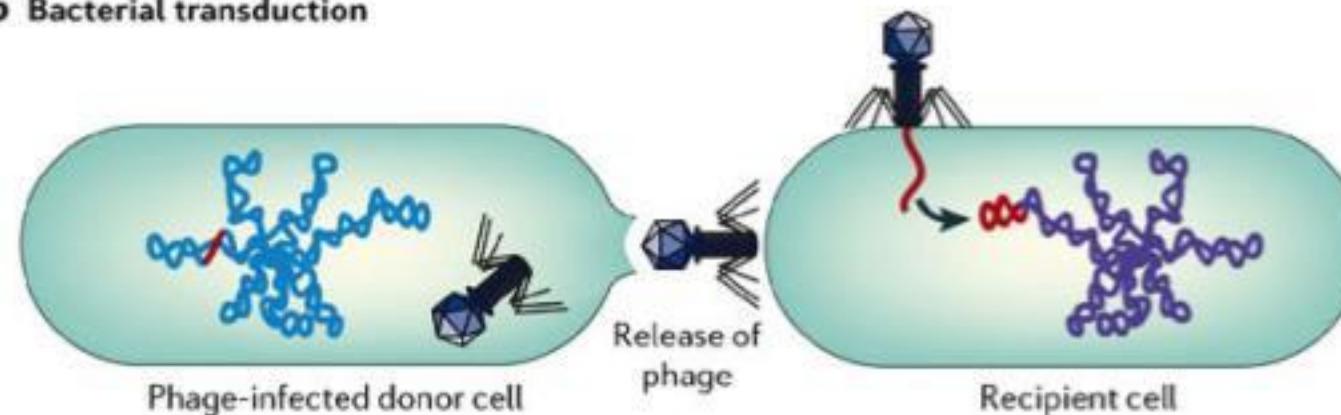
# Cómo intercambian genes en la naturaleza distintas especies

## a Bacterial transformation



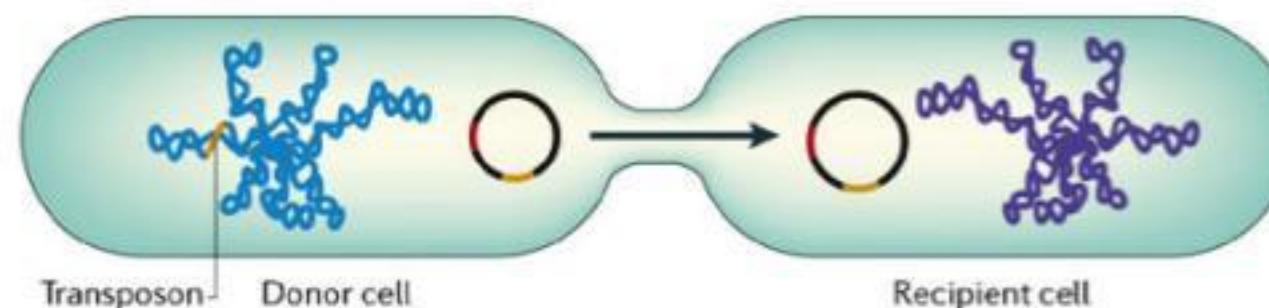
Transformación

## b Bacterial transduction



Transducción

## c Bacterial conjugation

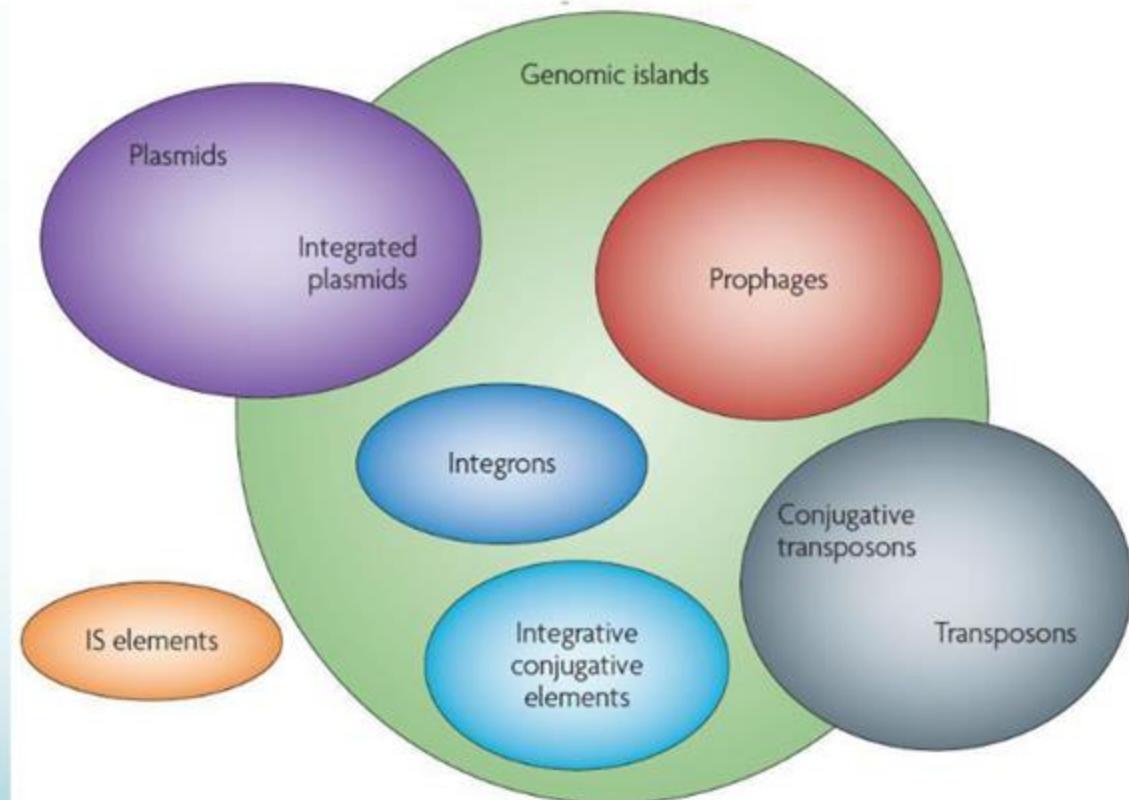


Conjugación



@victagua

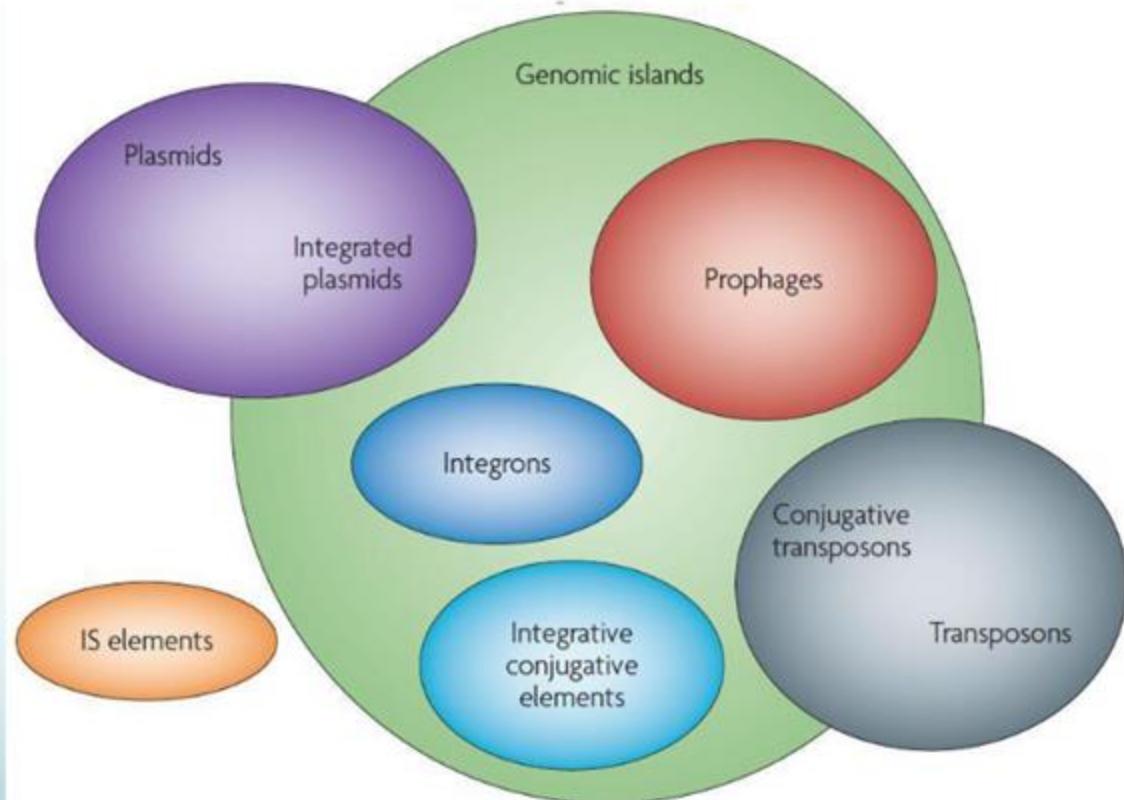
## Elementos Genéticos Móviles (EGMs)



**EGM** = Cualquier secuencia de ADN que se mueve físicamente dentro de un genoma de un organismo, o bien entre diferentes organismos.

*10 - 20% del genoma bacteriano son EGMs*

## Elementos Genéticos Móviles (EGMs): ISLAS GENÓMICAS (GIs)



Clúster de genes con evidencia de origen en la transferencia horizontal

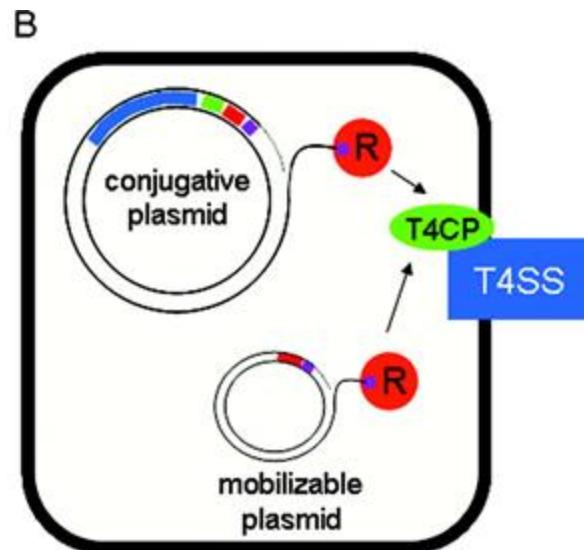
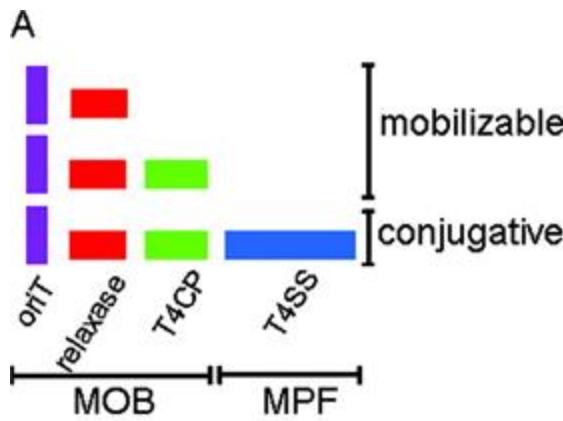
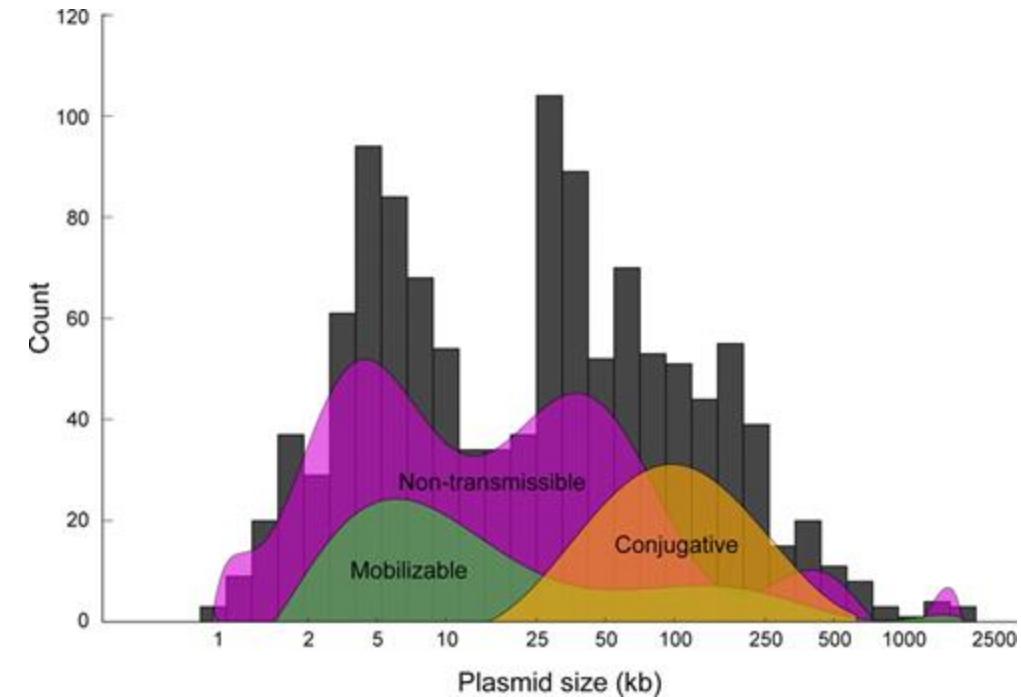
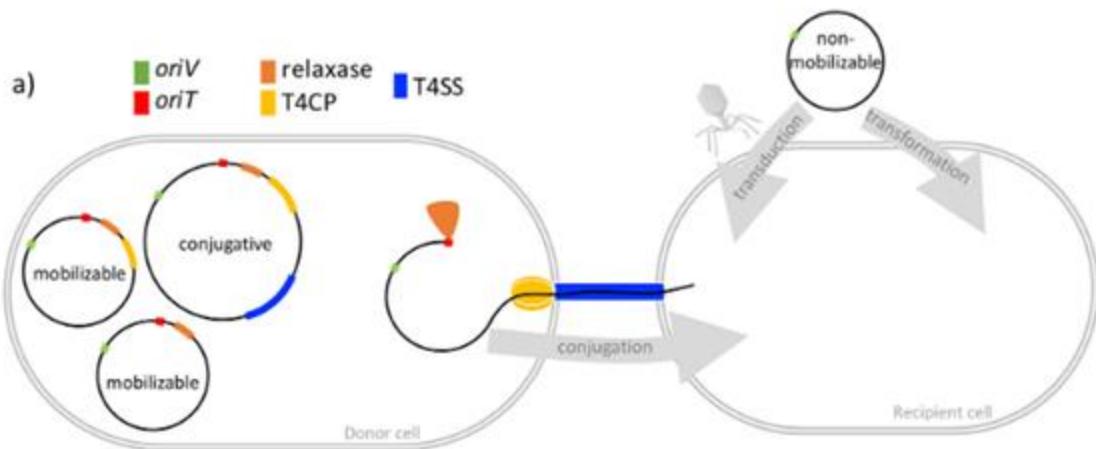
Contienen:

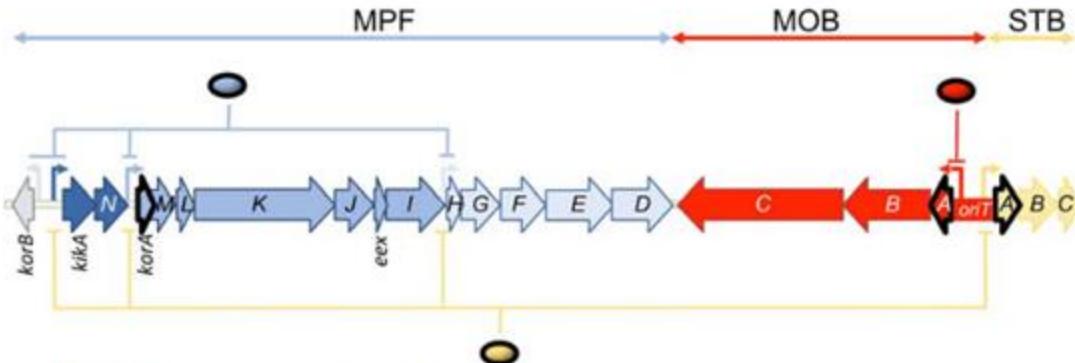
- profagos
- integrones
- Elementos integrativos conjugativos (ICEs)
- transposones conjugativos
- plásmidos integrados

Importancia:

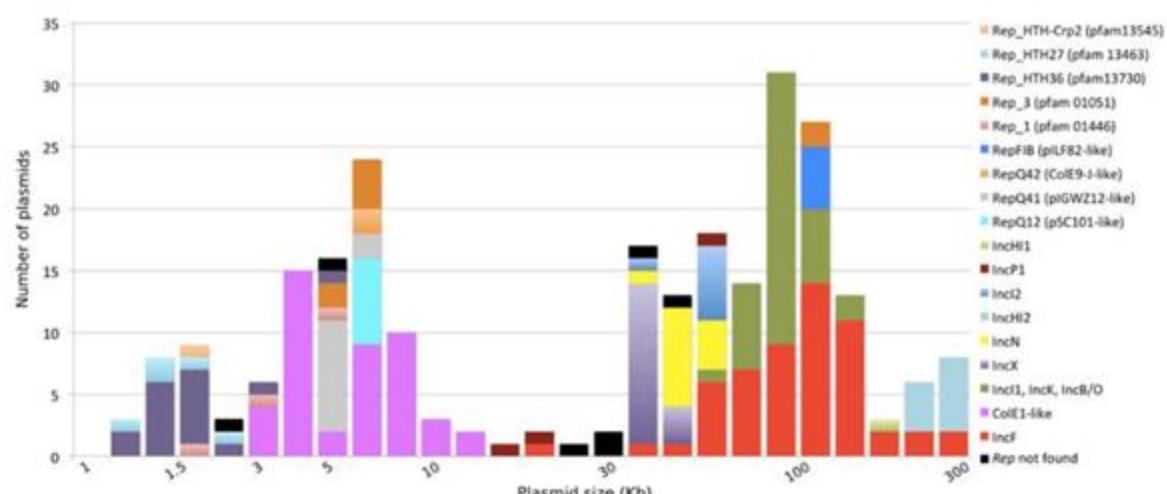
- Asociados con adaptación en ambientes seleccionados (médicos, ambientales, industriales), a través de resistencia a metales pesados, antibióticos, etc.
- Modular patogenicidad a través de factores de virulencia (frecuentemente sobrerepresentados en GIs)

# Elementos Genéticos Móviles (EGMs): PLÁSMIDOS



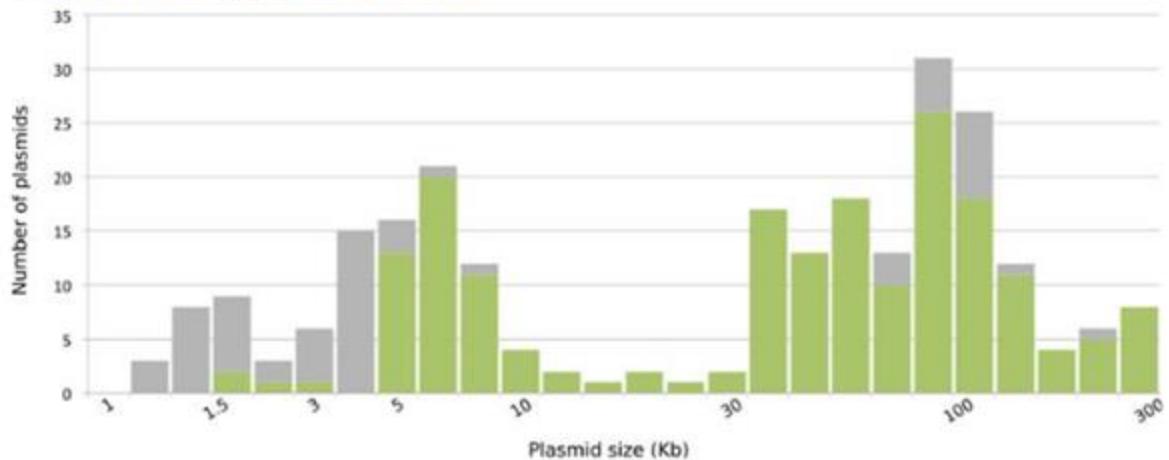


**FIGURE 1** Genetic map of the conjugation genes of plasmid R388. Three modules related to conjugative transfer are depicted: STB, MOB, and MPF. Genes encoded in the 18-kb fragment comprised between *korB* and *stbC* (GenBank Acc. No. BR000038) are represented by arrows. Genes contained in the same operon are depicted using the same color pattern, as well as the corresponding promoters, which are represented by small arrows. Genes encoding transcription factors are outlined in black. The encoded repressors follow the same color pattern as their corresponding genes and are represented as ovals. The activity of the repressors on the promoters is indicated by lines. doi:10.1128/microbiolspec.PLAS-0031-2014.f1

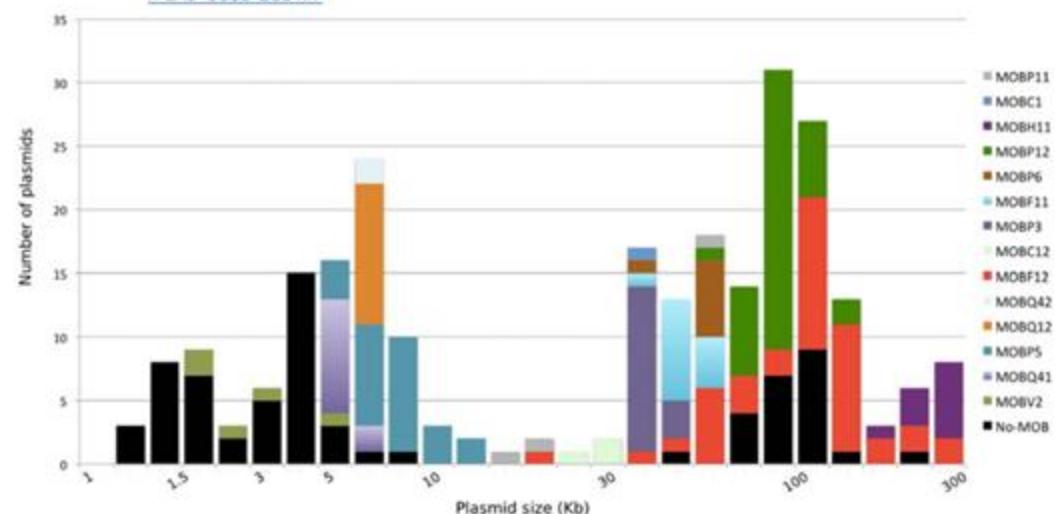


**FIGURE 6** Distribution of Inc/Rep types in the 255-*E. coli* plasmid collection. Plasmid REP types were identified as reported in references 73 and 147. doi:10.1128/microbiolspec.PLAS-0031-2014.f6

**FIGURE 2** Distribution of plasmid sizes in the 255-*E. coli* plasmid collection. The figure represents the size distribution of plasmids in the collection described in Table 1. The histogram shows the total number of plasmids corresponding to each size class (in logarithmic scale). Plasmid distribution shows a trimodal abundance curve (with calculated median sizes of 1.6, 5.1, and 87 kb). Plasmids in which a relaxase gene was detected are shown in green ( $N = 187$ ), and those in which it was not, in gray ( $N = 68$ ). doi:10.1128/microbiolspec.PLAS-0031-2014.f2

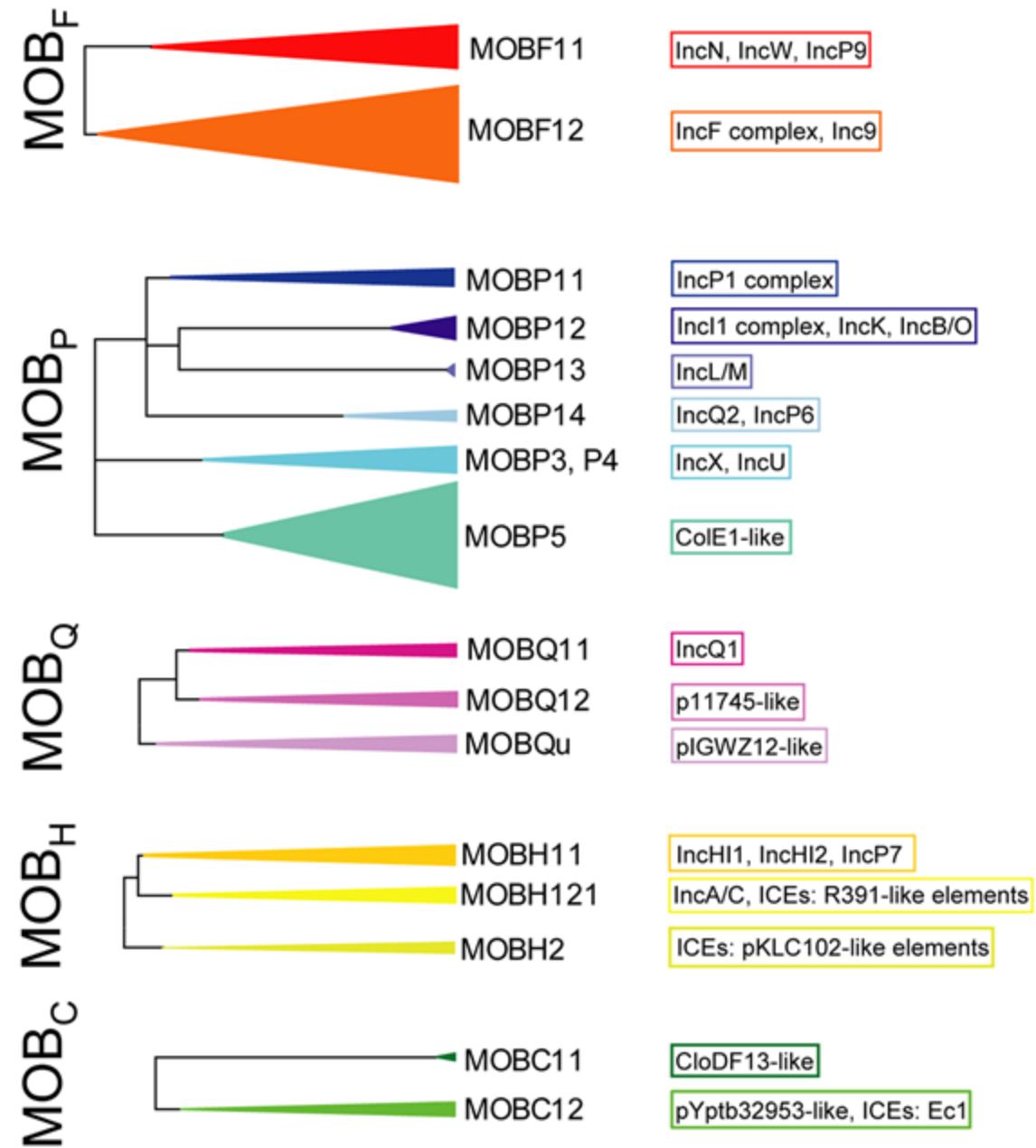


**FIGURE 7** Distribution of MOB types in the 255-*E. coli* plasmid collection. Plasmid MOB types were identified as reported in references 23 and 75. doi:10.1128/microbiolspec.PLAS-0031-2014.f7

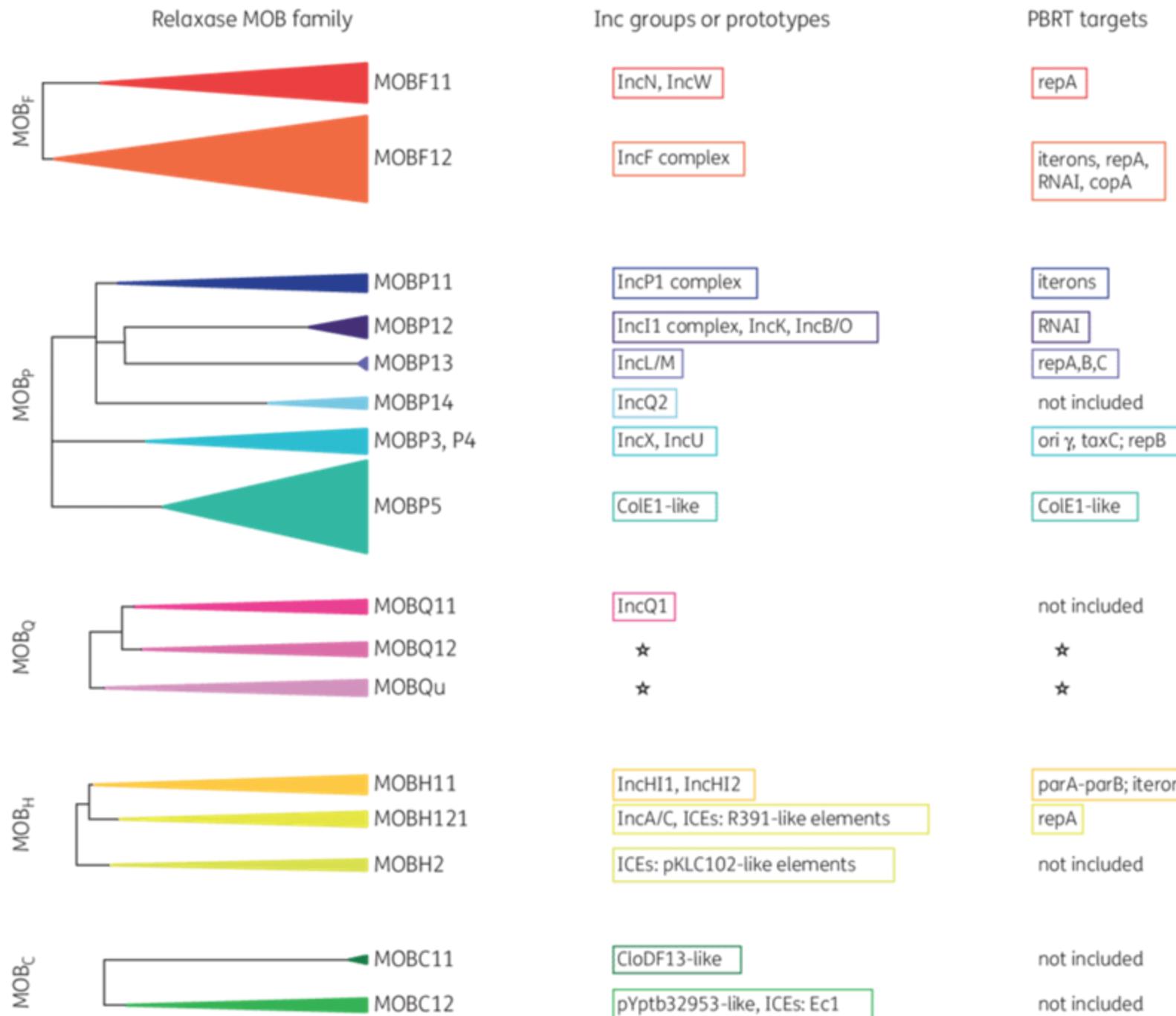


**FIGURE 6** Distribution of Inc/Rep types in the 255-*E. coli* plasmid collection. Plasmid REP types were identified as reported in references 73 and 147. doi:10.1128/microbiolspec.PLAS-0031-2014.f6

A. Relaxase MOB family



B. Inc groups or prototypes



## 5.6.1. Herramientas de ensamblaje y extracción de plásmidos

Estas son herramientas **desarrolladas para ensamblar y extraer los plásmidos directamente de las lecturas de secuenciación o los *contigs* ensamblados.**

En ellas se asume que los plásmidos tienen características diferenciales del cromosoma.

- Ejemplo de ello es asumir (no siempre de manera correcta) que los plásmidos están en mayor número de copias que el cromosoma, de manera que su cobertura al ser secuenciados es mayor que la del cromosoma.
- También se asume, que, debido a su naturaleza circular, el ensamblaje de los mismos puede circularizarse.

Herramientas que están en este grupo son cBar (Zhou y Xu, 2010), plasmidSPAdes (Antipov et al., 2016), Plasmid Profiler (Zetner et al., 2017), Recycler (Rozov et al., 2017), PlasmidSeeker (Roosaare et al., 2018), PlaScope (Royer et al., 2018), PlasFlow (Krawczyk et al., 2018), PlasmidTron (Page et al., 2018) o Platon (Schwengers et al., 2020).

## 5.6.1. Herramientas de ensamblaje y extracción de plásmidos

Las asunciones sobre las que se basan no siempre son ciertas.

- En primer lugar, solo los plásmidos multicopia presentan una cobertura media sustancialmente superior al cromosoma. De hecho, los plásmidos considerados grandes, que son principalmente aquellos que atraen nuestro interés por portar genes de resistencia a antibióticos, virulencia u otras funciones de interés epidemiológico, se suelen encontrar en una única copia respecto al cromosoma (De Toro et al., 2014; Fernandez-Lopez et al., 2017; Redondo-Salvo et al., 2020).
- Además, no debemos olvidar que el método de preparación de la genoteca puede afectar a la cobertura media final observada. En este sentido, los protocolos que conllevan un enriquecimiento por PCR de la genoteca pueden producir sesgos, enriqueciendo aquellas regiones o elementos que son más comunes en el genoma. Por este motivo, las librerías sin este tipo de enriquecimiento son deseables para la detección de EGM.
- Por otra parte, no todos los plásmidos son circulares, sino que existen (aunque son poco numerosos) plásmidos que son moléculas de ADN lineal, como aquellos encontrados en *Borrelia* spp., *Streptomyces* spp. o *Nocardia opaca*, por poner algunos ejemplos.

## 5.6.2. Herramientas de identificación mediante marcadores genéticos.

Localización de marcadores genéticos como son proteínas de replicación, mantenimiento, conjugación, etc.

- **PLACNET** (Lanza et al. 2014)/**PLACNETw** (Vielva et al., 2017): se combina el ensamblaje de los plásmidos, la información de cobertura y la visualización de los marcadores genéticos (RIP/REL).
- **PlasmidFinder** (Carattoli et al., 2014) o **MOBscan** (Garcillán-Barcia et al., 2020) actúan como base de datos y motor de búsqueda para regiones de replicación o relaxasa, respectivamente, permitiendo la tipificación de contigs o proteínas, tras el paso de ensamblaje.
- **MOBsuite** (Robertson & Nash, 2018) expande el esquema de tipado con la reconstrucción de las secuencias de los plásmidos a partir de ensamblajes.
- **mlplasmids** (Arredondo-Alonso et al, 2018), donde se aplica un sistema de clasificación binaria de especies para predecir si un contig deriva de un plásmido o de un cromosoma. Se realiza un entrenamiento previo mediante machine learning con genomas de referencia bien anotados.
- **COPLA** (Redondo-Salvo et al., 2021). Aplica índices de similitud nucleotídica (ANI), siendo capaz de clasificar los plásmidos en grupos taxonómicos (PTUs), previamente descritos en la base de datos (Redondo-Salvo et al., 2020).
- **MobileElementFinder** (<https://pypi.org/project/MobileElementFinder/>). Identificación de EGMs y su relación con genes de resistencia a antibióticos y factores de virulencia.
- Bases de datos con información sobre plásmidos y otros EGMs: ACLAME, plasmidATLAS, plasmidID...

### 5.6.3. Herramientas basadas en los grafos de ensamblaje de novo.

Estas herramientas no están especialmente diseñadas para el análisis de plásmidos, pero ha encontrado su utilidad al examinar los contigs y su relación entre ellos, para la detección de plásmidos y para integrar la información de lecturas cortas y largas.

Entre esas herramientas están:

- Contiguity (Sullivan et al., 2015),
- Bandage (Wick et al., 2015),
- Circlator (Hunt et al., 2015)
- Unicycler (Wick et al., 2017).



## Tema 5 - Ejemplo 7 - Elementos Genéticos Móviles

En este ejemplo vamos a realizar un ensamblaje diferencial de plásmidos utilizando la herramienta plasmidSPAdes. Revisaremos el grafo de ensamblaje obtenido con Bandage y tipificaremos los plásmidos mediante PlasmidFinder y COPLA. Para ello utilizaremos las lecturas del Ejemplo 1, obtenidos de Bioproject. que fueron filtrados y que se han utilizado en los ejemplos anteriores.

### Anotación con plasmidSPAdes:

Comando: `plasmidspades.py -1 out_1.clean.fastq.gz -2 out_2.clean.fastq.gz --careful -t 2 -k 127 -o plasmidSPAdes`.

Revisión de los archivos de salida: Lo primero que podemos notar al revisar el archivo contigs.fasta o scaffolds.fasta, es que las cabeceras de los contigs/scaffolds muestran una información adicional de “componente”, que se refiere a la distinción entre cromosoma o plásmidos realizada en base a la cobertura.

Preguntas: Recuenta el número de contigs y scaffolds obtenidos. ¿es consistente con el ensamblaje de genoma completo?

### Visualización del grafo de ensamblaje

Cargamos el archivo “assembly\_graph\_with\_scaffolds.gfa” que es el grafo de ensamblaje en Bandage.

Sacamos las estadísticas del ensamblaje en la opción: Graph information / More info

Dibujamos el grafo, con los parámetros por defecto: Grafo completo en estilo sencillo. A la vista de esta figura, ¿Cuántos plásmidos crees que podrías tener? ¿De qué tamaños y coberturas?

Podemos analizar lo que contiene cada uno de los nodos utilizando la función BLAST incluida en el programa (Output/Web BLAST selected nodes), que nos redireccionará a la página BLAST de NCBI. Con esta opción, realiza el BLAST de algunos de los componentes. ¿Qué información puedes obtener? ¿Puedes verificar que se trata de un plásmido?

### Tipificación de plásmidos mediante COPLA

Utilizamos el clasificador COPLA en su versión web (<https://castillo.dicom.unican.es/copla/>), donde incluiremos un identificador del trabajo y los scaffolds obtenidos del ensamblaje. Dado que conocemos la taxonomía de nuestro organismo, incluiremos la información apropiada al mismo ¿Qué tipo de plásmido nos indica que es? Busca información sobre ese tipo de PTU en la web de la base de datos ([https://castillo.dicom.unican.es/PlasmidID/RefSeq84\\_MOB/](https://castillo.dicom.unican.es/PlasmidID/RefSeq84_MOB/)).

Nota: Si quieras separar cada uno de los plásmidos en Bandage, pueden seleccionarse, y guardarse como un archivo FASTA en Output / Save selected node sequences to FASTA. A partir de los archivos FASTA, puede repetirse esta tipificación, para una mayor resolución.

### Tipificación de plásmidos mediante PlasmidFinder

En esta primera aproximación a PlasmidFinder utilizaremos su versión web, disponible en <https://cge.cbs.dtu.dk/services/PlasmidFinder/>, donde incluiremos los scaffolds o bien, los plásmidos separados previamente con Bandage en formato FASTA. Nota: seleccionad apropiadamente el tipo de organismo que estamos analizando (Gram positivo o Enterobacteriales). ¿Qué tipos de plásmidos localiza aquí?

Resultados del ejemplo [Tema5\\_Ejemplo7\\_plasmidos](#)

Comando:

```
plasmidspades.py -1 out_1.clean.fastq.gz -2 out_2.clean.fastq.gz --careful -t 2 -k 127 -o plasmidSPAdes
```

```
===== Terminate finished.

* Corrected reads are in /home/maria.detoro/04MBIF/Tema5_bacterianos/plasmidSPADES/corrected/
* Assembled contigs are in /home/maria.detoro/04MBIF/Tema5_bacterianos/plasmidSPADES/contigs.fasta
* Assembled scaffolds are in /home/maria.detoro/04MBIF/Tema5_bacterianos/plasmidSPADES/scaffolds.fasta
* Paths in the assembly graph corresponding to the contigs are in /home/maria.detoro/04MBIF/Tema5_bacterianos/plasmidSPADES/contigs.paths
* Paths in the assembly graph corresponding to the scaffolds are in /home/maria.detoro/04MBIF/Tema5_bacterianos/plasmidSPADES/scaffolds.paths
* Assembly graph is in /home/maria.detoro/04MBIF/Tema5_bacterianos/plasmidSPADES/assembly_graph.fasta
* Assembly graph in GFA format is in /home/maria.detoro/04MBIF/Tema5_bacterianos/plasmidSPADES/assembly_graph_with_scaffolds.gfa

===== SPAdes pipeline finished.

SPAdes log can be found here: /home/maria.detoro/04MBIF/Tema5_bacterianos/plasmidSPADES/spades.log
```

Comando:

```
plasmidspades.py -1 out_1.clean.fastq.gz -2 out_2.clean.fastq.gz --careful -t 2 -k  
127 -o plasmidSPAdes
```

### Revisión de los archivos de salida

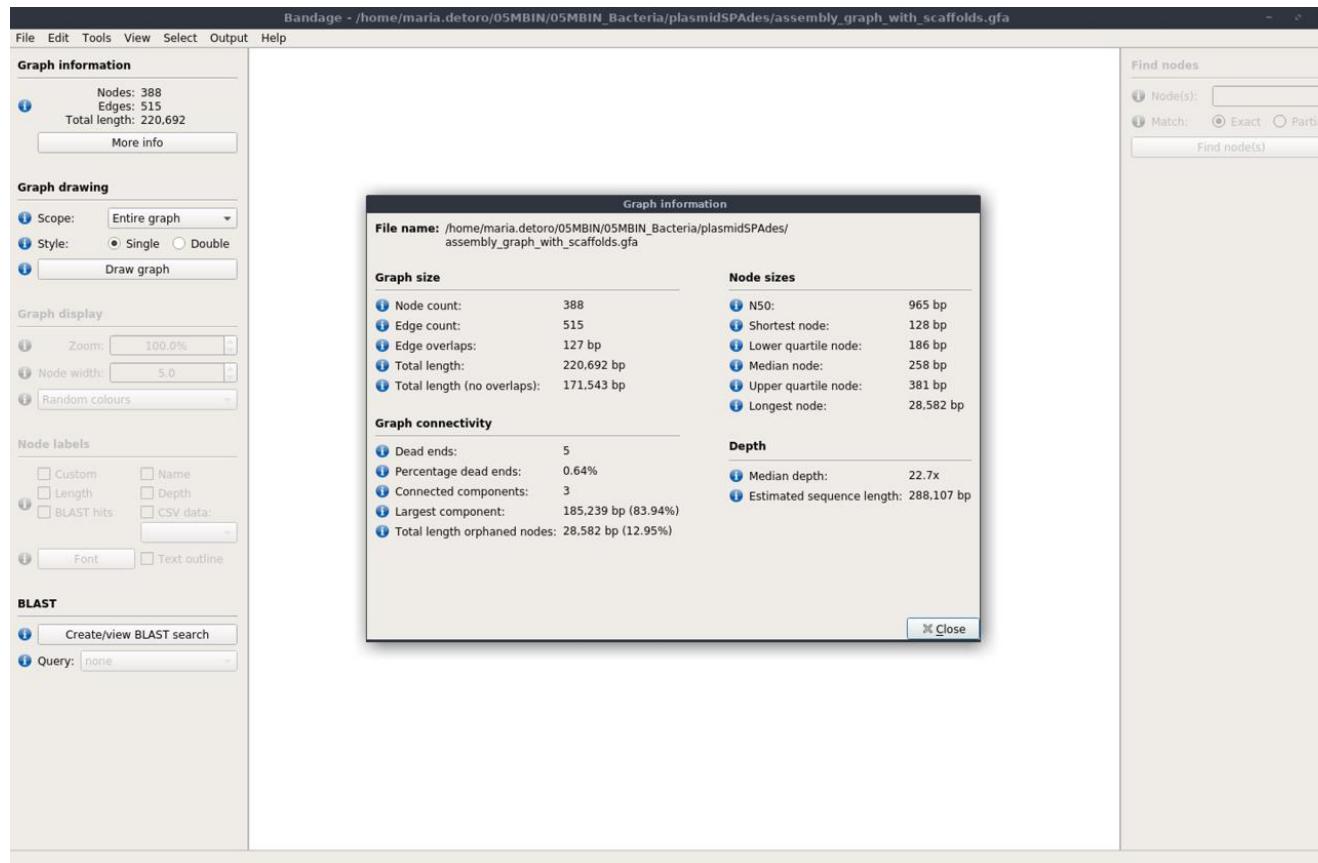
Lo primero que podemos notar al revisar el archivo *contigs.fasta* o *scaffolds.fasta*, es que las cabeceras de los *contigs/scaffolds* muestran una información adicional de “componente”, que se refiere a la distinción entre cromosoma o plásmidos realizada en función de la cobertura.

Recuenta el número de *contigs* y *scaffolds* obtenidos. ¿Es coherente con el ensamblaje de genoma completo?

## Visualización del grafo de ensamblaje

Cargamos el archivo assembly\_graph\_with\_scaffolds.gfa que es el grafo de ensamblaje en Bandage.

Sacamos las estadísticas del ensamblaje en la opción: *Graph information / More info*.



En los siguientes enlaces puedes encontrar más información sobre grafos y su visualización en Bandage.

<https://github.com/rrwick/Bandage/wiki>

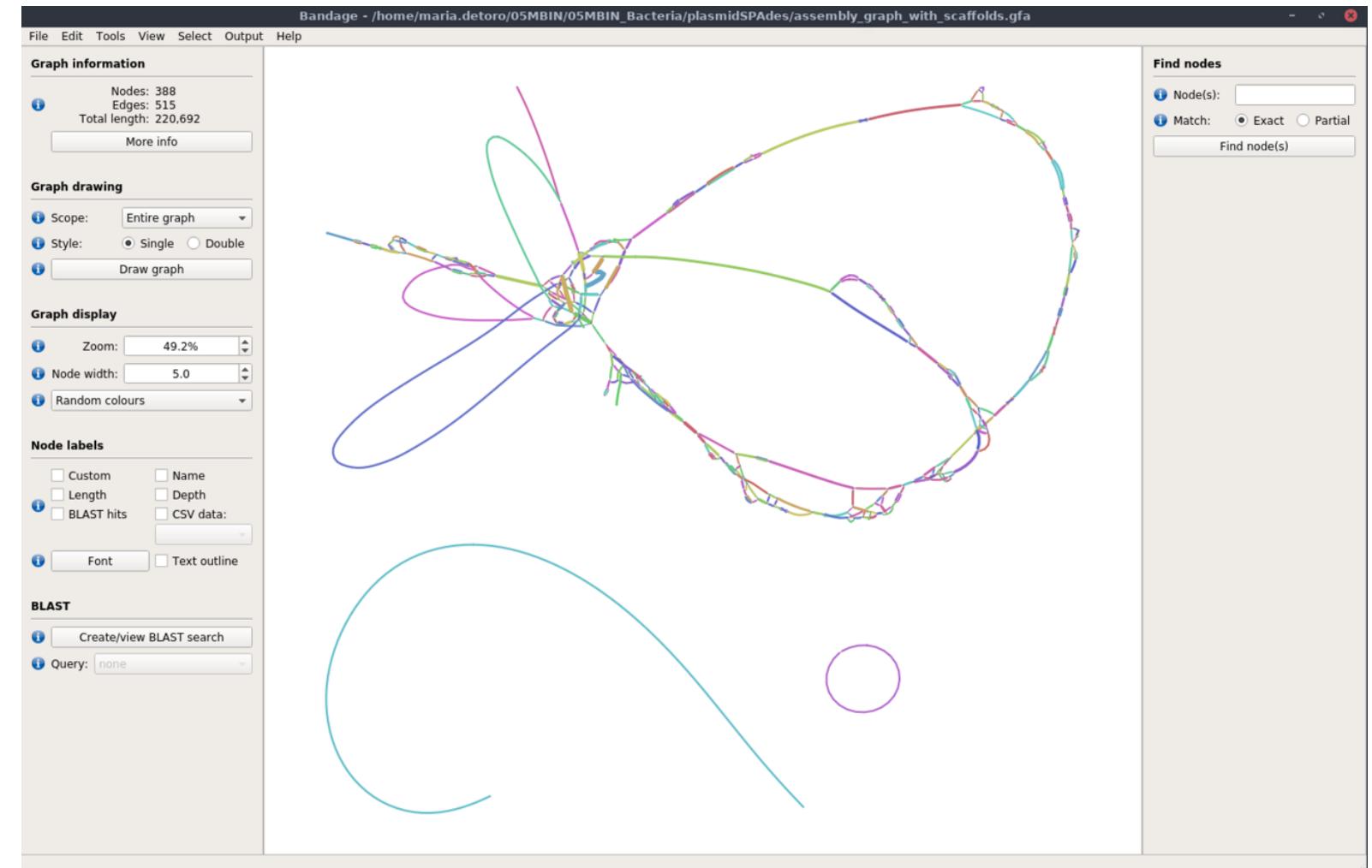
<https://github.com/rrwick/Bandage/wiki/Graph-paths>

<https://towardsdatascience.com/visualising-assembly-graphs-fb631f46bbd1>

<https://www.youtube.com/watch?v=cierloa5hS0>

Dibujamos el grafo, con los parámetros por defecto: grafo completo en estilo sencillo. A la vista de esta figura, ¿cuántos plásmidos crees que podrías tener? ¿De qué tamaños y coberturas?

Podemos analizar lo que contiene cada uno de los nodos utilizando la función BLAST incluida en el programa (*Output / Web BLAST selected nodes*), que nos redireccionará a la página BLAST de NCBI. Con esta opción, realiza el BLAST de algunos de los componentes. ¿Qué información puedes obtener? ¿Puedes verificar que se trata de un plásmido?





For each genome, upload the Illumina read files in the slots below, one for each fastq gzip-compressed read file.

As soon as your data is uploaded, we will provide you a link to the online processing tool. The data is expected to be ready in a few hours (you can refresh the link and check) and will remain accessible for one week.

Id of the job:

Read file 1 (.gz)  No se ha seleccionado ningún archivo.

Read file 2 (.gz)  No se ha seleccionado ningún archivo.

Please cite as: Luis Vielva, María de Toro, Val F Lanza, Fernando de la Cruz, PLACNETw: a web-based tool for plasmid reconstruction from bacterial genomes, *Bioinformatics*, Volume 33, Issue 23, 01 December 2017, Pages 3796-3798, <https://doi.org/10.1093/bioinformatics/btx462>

[Bibtex](#)

[Ris](#)

[Enw](#)

[Tutorial video](#)

[Source code](#)

[Examples](#)

Incluye ensamblaje (Velvet, opción SPAdes bajo demanda)

Limpieza manual de la red → necesario conocimiento sobre plásmidos!

*Bioinformatics*, 33(23), 2017, 3796–3798

doi: 10.1093/bioinformatics/btx462

Advance Access Publication Date: 21 July 2017

Applications Note



## Genome analysis

# PLACNETw: a web-based tool for plasmid reconstruction from bacterial genomes

Luis Vielva<sup>1</sup>, María de Toro<sup>2</sup>, Val F. Lanza<sup>3</sup> and Fernando de la Cruz<sup>4,\*</sup>

<sup>1</sup>Departamento de Ingeniería de Comunicaciones, Universidad de Cantabria, Santander, Spain, <sup>2</sup>Plataforma de Genómica y Bioinformática, Centro de Investigación Biomédica de La Rioja (CIBIR), Logroño, Spain,

<sup>3</sup>Departamento de Microbiología, Instituto de Investigación Sanitaria Ramón y Cajal (IRYCIS), Madrid, Spain and

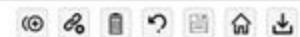
<sup>4</sup>Departamento de Biología Molecular, Universidad de Cantabria and Instituto de Biomedicina y Biotecnología de Cantabria (Universidad de Cantabria-CSIC), Santander, Spain

\*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on October 24, 2016; revised on June 20, 2017; editorial decision on July 13, 2017; accepted on July 19, 2017

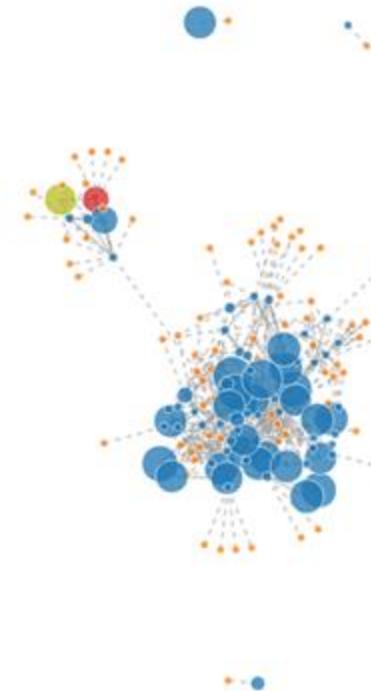
## Node tree



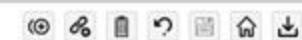
- > ● Node: 1, Length: 130599, Covera
- > ● Node: 2, Length: 41217, Covera
- > ● Node: 3, Length: 186605, Covera
- > ● Node: 4, Length: 339554, Covera
- > ● Node: 5, Length: 91428, Covera
- > ● Node: 6, Length: 398544, Covera
- > ● Node: 7, Length: 15590, Covera
- > ● Node: 8, Length: 111281, Covera
- > ● Node: 9, Length: 180744, Covera
- > ● Node: 10, Length: 1725, Covera
- > ● Node: 11, Length: 76654, Covera
- > ● Node: 12, Length: 17462, Covera
- > ● Node: 13, Length: 16781, Covera
- > ● Node: 14, Length: 1911, Covera
- > ● Node: 15, Length: 41388, Covera
- > ● Node: 16, Length: 434506, Covera
- > ● Node: 17, Length: 25087, Covera
- > ● Node: 18, Length: 179162, Covera
- > ● Node: 19, Length: 29405, Covera
- > ● Node: 20, Length: 112126, Covera
- > ● Node: 21, Length: 108305, Covera
- > ● Node: 22, Length: 50887, Covera
- > ● Node: 23, Length: 54217, Covera
- > ● Node: 24, Length: 53961, Covera
- > ● Node: 25, Length: 29837, Covera
- > ● Node: 26, Length: 671038, Covera
- > ● Node: 27, Length: 308089, Covera
- > ● Node: 28, Length: 4285, Covera
- > ● Node: 29, Length: 1189, Covera
- > ● Node: 30, Length: 26462, Covera
- > ● Node: 31, Length: 701, Coverag

Additional info Notes Log

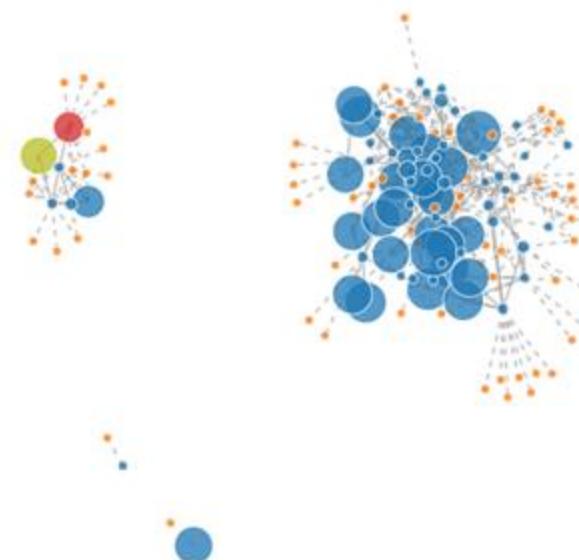
You can take your own notes here...



## Node tree



- > ● Node: 1, Length: 130599, Covera
- > ● Node: 2, Length: 41217, Covera
- > ● Node: 3, Length: 186605, Covera
- > ● Node: 4, Length: 339554, Covera
- > ● Node: 5, Length: 91428, Covera
- > ● Node: 6, Length: 398544, Covera
- > ● Node: 7, Length: 15590, Covera
- > ● Node: 8, Length: 111281, Covera
- > ● Node: 9, Length: 181019, Covera
- > ● Node: 10, Length: 1725, Covera
- > ● Node: 11, Length: 76654, Covera
- > ● Node: 12, Length: 17462, Covera
- > ● Node: 13, Length: 16781, Covera
- > ● Node: 14, Length: 41388, Covera
- > ● Node: 15, Length: 434506, Covera
- > ● Node: 16, Length: 25087, Covera
- > ● Node: 17, Length: 179162, Covera
- > ● Node: 18, Length: 29405, Covera
- > ● Node: 19, Length: 112126, Covera
- > ● Node: 20, Length: 108425, Covera
- > ● Node: 21, Length: 50887, Covera
- > ● Node: 22, Length: 54217, Covera
- > ● Node: 23, Length: 53961, Covera
- > ● Node: 24, Length: 29837, Covera
- > ● Node: 25, Length: 671038, Covera
- > ● Node: 26, Length: 308089, Covera
- > ● Node: 27, Length: 4285, Covera



Additional info Notes Log

```

23/4/2017 19:41:27: Network saved
23/4/2017 19:41:20: Delete link from NODE_35_length_677_cov_168.871490 to gi|387615344|ref|NC_017634.1|
23/4/2017 19:41:11: Network loaded
23/4/2017 19:33:23: Network saved
23/4/2017 19:32:56: Network saved
23/4/2017 19:32:46: Network loaded
23/4/2017 19:20:30: Network saved
23/4/2017 19:20:24: Delete link from NODE_35_length_677_cov_168.871490 to gi|387615344|ref|NC_017634.1|
23/4/2017 19:19:10: Network loaded
23/4/2017 19:18:51: Network saved
23/4/2017 19:17:54: Delete link from NODE_35_length_677_cov_168.871490 to gi|387615344|ref|NC_017634.1|
23/4/2017 19:17:54: Delete node gi|387615344|ref|NC_017634.1|
23/4/2017 19:17:22: Selected contigs by node length from 200 to 5000
Network loaded

```

# PlasmidFinder and In Silico pMLST: Identification and Typing of Plasmid Replicons in Whole-Genome Sequencing (WGS)

Alessandra Carattoli<sup>1</sup>, Henrik Hasman<sup>2</sup>

Affiliations + expand

PMID: 31584170 DOI: 10.1007/978-1-4939-9877-7\_20

<https://cge.food.dtu.dk/services/PlasmidFinder/>

<https://bitbucket.org/genomicepidemiology/plasmidfinder/src/master/>

The screenshot shows the homepage of the Center for Genomic Epidemiology. The header includes the institution's name and a navigation bar with links for Home, Services, Publications, and Contact. Below the header, the PlasmidFinder service is highlighted. It features a sub-header "PlasmidFinder 2.1", tabs for Service, Instructions, Output, Article abstract, and Citations. Under the Service tab, it displays software version 2.0.1 (2020-07-01) and database version (2021-11-29). A "Test sequence" button is present. On the right, a yellow box states: "The database is curated by: Henrik Hasman and Alessandra Carattoli (click to contact)". A sidebar titled "Select database" offers options for Gram Positive and Enterobacteriales. Other settings include "Select threshold for minimum % identity" (95 %) and "Select minimum % coverage" (60 %). At the bottom, there is a note about sequencing reads and a dropdown menu for genome type.

The screenshot shows a GitHub repository page for "PlasmidFinder". The top navigation bar shows the repository path "Genomic Epidemiology / COE PlasmidFinder" and includes a dropdown for "master", a "Files" dropdown, a "Filter files" input, and a search icon. The main content is a file list table with columns for Name, Size, Last commit, and Message. The table contains the following entries:

Name	Size	Last commit	Message
test		2021-03-27	Removed duplicated test repository
.gitignore	34 B	2015-07-29	Update
Dockerfile	1.13 KB	2021-03-09	Change spelling error "pasmidfinder.py" to "plasmidfinder.py" in ENTRYPOINT command
README.md	4.94 KB	2022-05-12	Adds usage comment in README.md from help.
plasmidFinder.py	20.33 KB	2020-02-07	Fixed bug

Below the file list, there is a section titled "README.md" which contains the following content:

## PlasmidFinder

This project documents the PlasmidFinder service

## Documentation

The PlasmidFinder service contains one python script `plasmidFinder.py` which is the script of the latest version of the PlasmidFinder service. The service identifies plasmids in total or partial sequenced isolates of bacteria.

PlasmidFinder allows identification of plasmids in total or partial sequenced isolates of bacteria.

[Conda](#)[Files](#)[Labels](#)[Badges](#)

 License: [Apache-2.0](#)

 Home: <https://bitbucket.org/genomicepidemiology/plasmidfinder>

 2879 total downloads

 Last upload: 5 days and 12 hours ago

## Installers

`conda install` 

   noarch v2.1.6

To install this package with conda run:

```
conda install -c bioconda plasmidfinder
```

## Description

# homemade = BLAST + DB PlasmidFinder

## 1) Descargar la base de datos del repositorio

```
git clone https://bitbucket.org/genomicepidemiology/plasmidfinder_db.git
```

```
(MGE) pgen@pgen:/media/pgen/Disco_2/curso_leon/reads_clean/contigs/plasmidfinder_db$ ll -h
total 440K
drwxrwxr-x  3 pgen pgen 4,0K jun 16 13:30 .
drwxrwxr-x 10 pgen pgen 4,0K jun 16 13:30 ..
-rw-rw-r--  1 pgen pgen  469 jun 16 13:30 config
-rw-rw-r--  1 pgen pgen  70K jun 16 13:30 enterobacteriales.fsa
drwxrwxr-x  8 pgen pgen 4,0K jun 16 13:30 .git/
-rw-rw-r--  1 pgen pgen   11 jun 16 13:30 .gitignore
-rw-rw-r--  1 pgen pgen  60K jun 16 13:30 Inc18.fsa
-rwxrwxr-x  1 pgen pgen 1,9K jun 16 13:30 INSTALL.py*
-rw-rw-r--  1 pgen pgen  9,1K jun 16 13:30 NT_Rep.fsa
-rw-rw-r--  1 pgen pgen  65K jun 16 13:30 Rep1.fsa
-rw-rw-r--  1 pgen pgen  15K jun 16 13:30 Rep2.fsa
-rw-rw-r--  1 pgen pgen  53K jun 16 13:30 Rep3.fsa
-rw-rw-r--  1 pgen pgen  86K jun 16 13:30 RepA_N.fsa
-rw-rw-r--  1 pgen pgen  7,7K jun 16 13:30 RepL.fsa
-rw-rw-r--  1 pgen pgen  36K jun 16 13:30 Rep_trans.fsa
```

```
# Database configuration file - Describes the content of the database
# Each db consist of 1 file(s) with the following extensions: fsa
# Other important files are: notes.txt
#db_prefix      name      description
enterobacteriales      Enterobacteriales      Enterobacteriales
Inc18      Gram Positive      Inc18
NT_Rep      Gram Positive      NT_Rep
Rep1       Gram Positive      Rep1
Rep2       Gram Positive      Rep2
Rep3       Gram Positive      Rep3
RepA_N     Gram Positive      RepA_N
RepL       Gram Positive      RepL
Rep_trans    Gram Positive      Rep_trans
config (END)
```

# homemade = BLAST + DB PlasmidFinder

## 1) Indexamos la base de datos a utilizar

```
(MGE) pgen@pgen:/media/pgen/Disco_2/curso_leon/reads_clean/contigs/plasmidfinder_db$ makeblastdb -in enterobacteriales.fsa -dbtype nucl
```

```
Building a new DB, current time: 06/16/2022 13:35:25
New DB name: /media/pgen/Disco_2/curso_leon/reads_clean/contigs/plasmidfinder_db/enterobacteriales.fsa
New DB title: enterobacteriales.fsa
Sequence type: Nucleotide
Keep MBits: T
Maximum file size: 10000000000B
Adding sequences from FASTA; added 142 sequences in 0.00216889 seconds.
```

```
(MGE) pgen@pgen:/media/pgen/Disco_2/curso_leon/reads_clean/contigs/plasmidfinder_db$ ll -h
total 520K
drwxrwxr-x 3 pgen pgen 4,0K jun 16 13:35 .
drwxrwxr-x 10 pgen pgen 4,0K jun 16 13:30 ../
-rw-rw-r-- 1 pgen pgen 469 jun 16 13:30 config
-rw-rw-r-- 1 pgen pgen 70K jun 16 13:30 enterobacteriales.fsa
-rw-rw-r-- 1 pgen pgen 20K jun 16 13:35 enterobacteriales.fsa.ndb
-rw-rw-r-- 1 pgen pgen 12K jun 16 13:35 enterobacteriales.fsa.nhr
-rw-rw-r-- 1 pgen pgen 1,8K jun 16 13:35 enterobacteriales.fsa.nin
-rw-rw-r-- 1 pgen pgen 1,7K jun 16 13:35 enterobacteriales.fsa.not
-rw-rw-r-- 1 pgen pgen 17K jun 16 13:35 enterobacteriales.fsa.nsq
-rw-rw-r-- 1 pgen pgen 16K jun 16 13:35 enterobacteriales.fsa.ntf
-rw-rw-r-- 1 pgen pgen 572 jun 16 13:35 enterobacteriales.fsa.nto
drwxrwxr-x 8 pgen pgen 4,0K jun 16 13:30 .git/
-rw-rw-r-- 1 pgen pgen 11 jun 16 13:30 .gitignore
-rw-rw-r-- 1 pgen pgen 60K jun 16 13:30 Inc18.fsa
-rwxrwxr-X 1 pgen pgen 1,9K jun 16 13:30 INSTALL.py*
-rw-rw-r-- 1 pgen pgen 9,1K jun 16 13:30 NT_Rep.fsa
-rw-rw-r-- 1 pgen pgen 65K jun 16 13:30 Rep1.fsa
-rw-rw-r-- 1 pgen pgen 15K jun 16 13:30 Rep2.fsa
-rw-rw-r-- 1 pgen pgen 53K jun 16 13:30 Rep3.fsa
-rw-rw-r-- 1 pgen pgen 86K jun 16 13:30 RepA_N.fsa
-rw-rw-r-- 1 pgen pgen 7,7K jun 16 13:30 RepL.fsa
-rw-rw-r-- 1 pgen pgen 36K jun 16 13:30 Rep_trans.fsa
```

# homemade = BLAST + DB PlasmidFinder

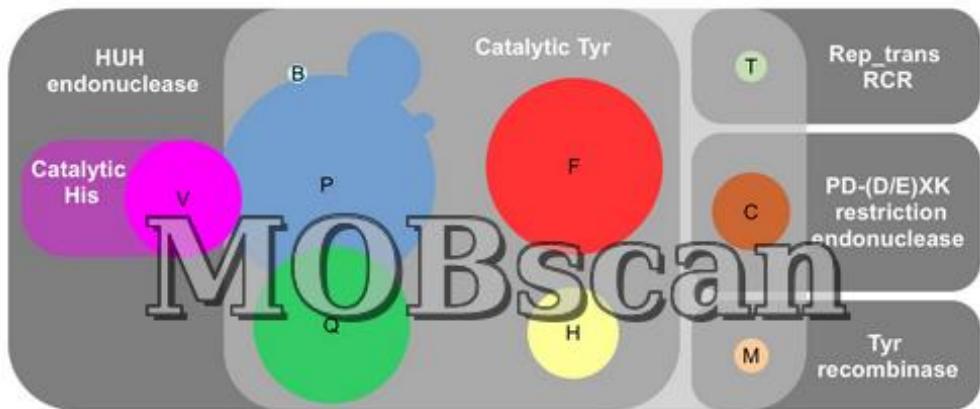
## 2) Ejecutamos blastn

```
blastn -query contigs/FC2.fasta -db plasmidfinder_db/enterobacteriales.fsa -evalue 1e-10 -outfmt 6 -out FC2_plasmidfinder.csv
```

(MGE) pgen@pgen:/media/pgen/Disco_2/curso_leon/reads_clean/contigs\$	blastn -query contigs/FC2.fasta -db plasmidfinder_db/enterobacteriales.fsa -evalue 1e-10 -outfmt 6
NODE_28_length_26023_cov_5.208517	IncFIA_1_AP001918 100.000 388 0 0 13090 13477 1 388 0.0 717
NODE_28_length_26023_cov_5.208517	IncFIA(HI1)_1_AF250878 88.235 391 41 3 13089 13477 1 388 8.22e-131 462
NODE_28_length_26023_cov_5.208517	FIA(pBK30683)_1_KF954760 91.065 291 26 0 13187 13477 5 295 3.04e-110 394
NODE_51_length_6274_cov_5.119961	IncFII(pRSB107)_1_AJ851089 100.000 261 0 0 1234 1494 261 1 1.52e-137 483
NODE_51_length_6274_cov_5.119961	IncFII(pAMA1167-NDM-5)_1_CP024805 100.000 238 0 0 1257 1494 238 1 9.28e-125 440
NODE_51_length_6274_cov_5.119961	IncFII(pHN7A8)_1_JN232517 95.402 261 11 1 1234 1494 260 1 5.63e-117 414
NODE_51_length_6274_cov_5.119961	IncFII(pSF0)_1_AF401292 93.870 261 13 3 1234 1494 1 258 9.48e-110 390
NODE_51_length_6274_cov_5.119961	IncFII(29)_1_CP003035 90.458 262 21 3 1234 1494 259 1 2.69e-95 342
NODE_51_length_6274_cov_5.119961	IncFIC(FII)_1_AP001918 89.051 274 28 2 1222 1494 273 1 3.48e-94 339
NODE_51_length_6274_cov_5.119961	IncFII(pSE11)_1_AP009242 89.434 265 23 3 1234 1494 264 1 2.10e-91 329
NODE_51_length_6274_cov_5.119961	IncFII_1_AY458016 88.550 262 28 2 1234 1494 261 1 1.63e-87 316
NODE_51_length_6274_cov_5.119961	IncFII(pCoo)_1_CR942285 87.072 263 31 3 1234 1494 262 1 7.65e-81 294
NODE_51_length_6274_cov_5.119961	IncFII(p14)_1_JQ418538 85.660 265 26 8 1241 1496 1 262 4.63e-73 268
NODE_51_length_6274_cov_5.119961	IncFII(Yp)_1_CP000670 85.652 230 27 5 1275 1499 1 229 1.31e-63 237
NODE_51_length_6274_cov_5.119961	FII(Cf)_1_CP056911 79.848 263 42 9 1242 1498 1 258 6.21e-47 182
NODE_51_length_6274_cov_5.119961	IncFII(S)_1_CP000858 79.762 252 40 11 1243 1486 2 250 3.74e-44 172
NODE_51_length_6274_cov_5.119961	FII(pBK30683)_1_KF954760 77.155 232 41 8 1275 1499 1 227 1.06e-29 124

When not provided, the default value is:  
'qaccver saccver pidlength mismatch gapopen qstart qend sstart send  
evalue bitscore', which is equivalent to the keyword 'std'

```
mkdir plasmidfinder
plasmidFinder_DB= ./plasmidfinder_db/enterobacteriales.fsa
for i in contigs/*.fasta; do
filename=$(basename ${i%.*}.fasta)
blastn -query contigs/${filename}.fasta -db ${plasmidFinder_DB} -evalue 1e-10
-outfmt 6 -out plasmidfinder/${filename}_plasmidFinder.csv
done
```



MOBscan is a web application for identifying relaxase MOB families. It uses the hmmscan function of the HMMER3 software suite ([Eddy, 2011](#)) to search against [MOBfamDB](#), a curated relaxase profile HMM database. If you find it useful for your work, please cite it as:

Garcillán-Barcia M.P., Redondo-Salvo S., Vielva L., de la Cruz F. (2020) "MOBscan: Automated Annotation of MOB Relaxases". In: de la Cruz F. (eds) *Horizontal Gene Transfer. Methods in Molecular Biology*, vol 2075. Humana, New York, NY

## MOBfamDB construction information

Individual profile HMM files for relaxase families MOB<sub>P</sub> (T4SS\_MOBP1, T4SS\_MOBP2, and T4SS\_MOBP3), MOB<sub>Q</sub> (T4SS\_MOBQ), MOB<sub>H</sub> (T4SS\_MOBH), MOB<sub>C</sub> (T4SS\_MOBC), MOB<sub>V</sub> (T4SS\_MOBV), and MOB<sub>B</sub> (T4SS\_MOBB) have been retrieved from the [TXSScan](#) suite ([Abby et al., 2016](#)).

New profile HMMs were built for MOB<sub>F</sub>, MOB<sub>T</sub>, and MOB<sub>M</sub> families from seed multiple sequence alignments generated with MUSCLE v3.8.31 ([Edgar, 2004](#)) by using the hmmbuild function of HMMER3.

- **MOB<sub>F</sub> Profile**

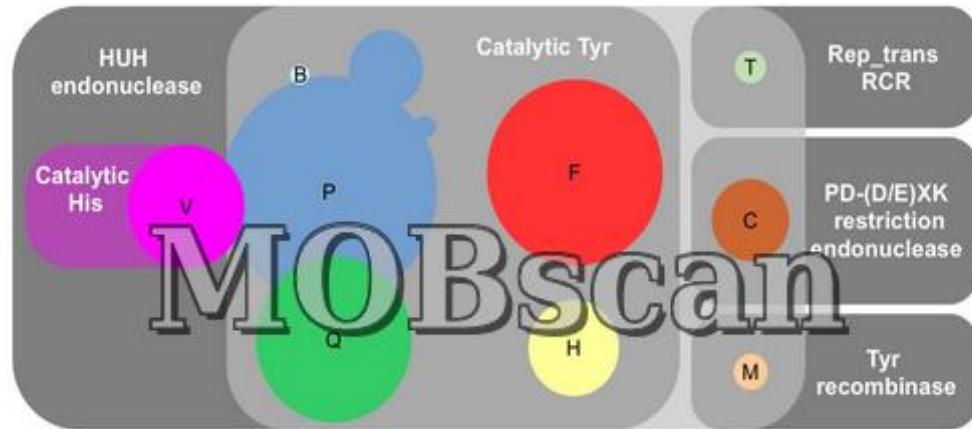
The profile HMM for the MOB<sub>F</sub> relaxase family (profile\_MOBF) was generated from an [alignment](#) of the N-terminal 300 residues of 146 MOB<sub>F</sub> proteins described by ([Garcillán-Barcia et al., 2009](#)), which was previously trimmed to remove poorly aligned regions at the edges by using Trimal ([Capella-Gutiérrez et al., 2009](#)).

- **MOB<sub>T</sub> Profile**

The profile HMM for the MOB<sub>T</sub> relaxase family (profile\_MOBT) was built from an [alignment](#) of 498 relaxases retrieved from a BLASTP search ([Altschul et al., 1990](#)) using as query the Orf20\_Tn916 complete sequence present in Figure 3 of ([Wright and Grossman, 2016](#)). The alignment was trimmed while keeping both HTH and Rep\_trans domains.

- **MOB<sub>M</sub> Profile**

For the novel family MOB<sub>M</sub>, a profile HMM (profile\_MOBM) was built based on an [alignment](#) of thirteen relaxases from *Clostridium perfringens* plasmids, retrieved from a BLASTP search using TcpM\_pCW3 (GenBank Acc. No. [WP\\_003479716.1](#)) as query.



Please, fill the submission form below and select the FASTA file containing the amino acid sequences.

Id of the job:

FASTA file (.faa)

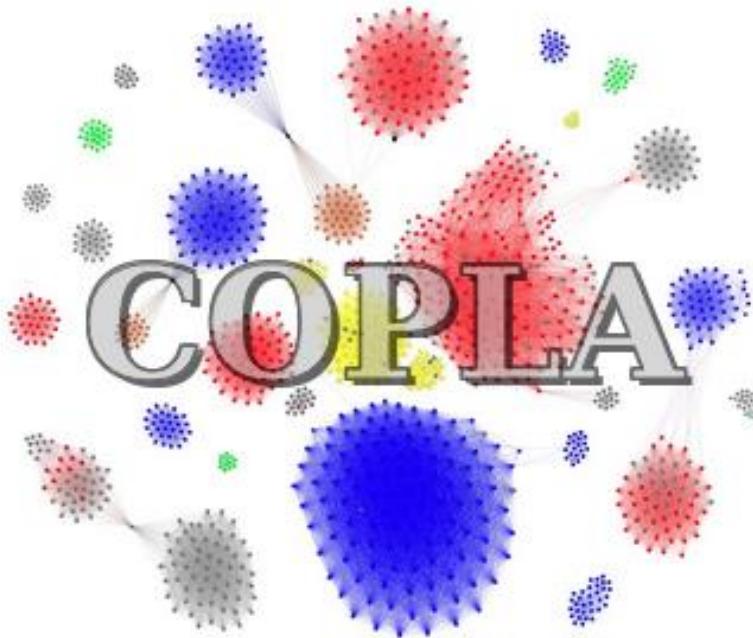
No se ha seleccionado ningún archivo.

[CONJScan](#)

[TXSScan](#)

[HMMER search](#)

[About MOBscan](#)



COPLA is an application for plasmid classification in Plasmid Taxonomic Units (PTUs). This web service is an online implementation of COPLA. If you find it useful for your work, please cite:

Redondo-Salvo, S., Bartomeus, R., Vielva, L., Tagg, K.A., Webb, H.E., Fernandez-Lopez, R., de la Cruz, F. (2020) "COPLA, a taxonomic classifier of plasmids". *bioRxiv*

Redondo-Salvo, S., Fernandez-Lopez, R., Ruiz., R., Vielva, L., de Toro, M., Rocha, E.P.C., Garcillan-Barcia, M.P., de la Cruz, F. (2020) "Pathways for Horizontal Gene Transfer in Bacteria Revealed by a Global Map of Their Plasmids". *Nat Commun* **11**, 3602



## ¿Qué son los PTUs? Plasmid Taxonomic Units

### nature communications

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [nature communications](#) > [articles](#) > [article](#)

Article | [Open Access](#) | Published: 17 July 2020

### Pathways for horizontal gene transfer in bacteria revealed by a global map of their plasmids

[Santiago Redondo-Salvo](#), [Raúl Fernández-López](#), [Raúl Ruiz](#), [Luis Vielva](#), [María de Toro](#), [Eduardo P. C. Rocha](#), [M. Pilar Garcillán-Barcia](#) & [Fernando de la Cruz](#)✉

*Nature Communications* **11**, Article number: 3602 (2020) | [Cite this article](#)

21k Accesses | 99 Citations | 124 Altmetric | [Metrics](#)

Redondo-Salvo et al. *BMC Bioinformatics* (2021) 22:390  
<https://doi.org/10.1186/s12859-021-04299-x>

BMC Bioinformatics

SOFTWARE

Open Access

### COPLA, a taxonomic classifier of plasmids

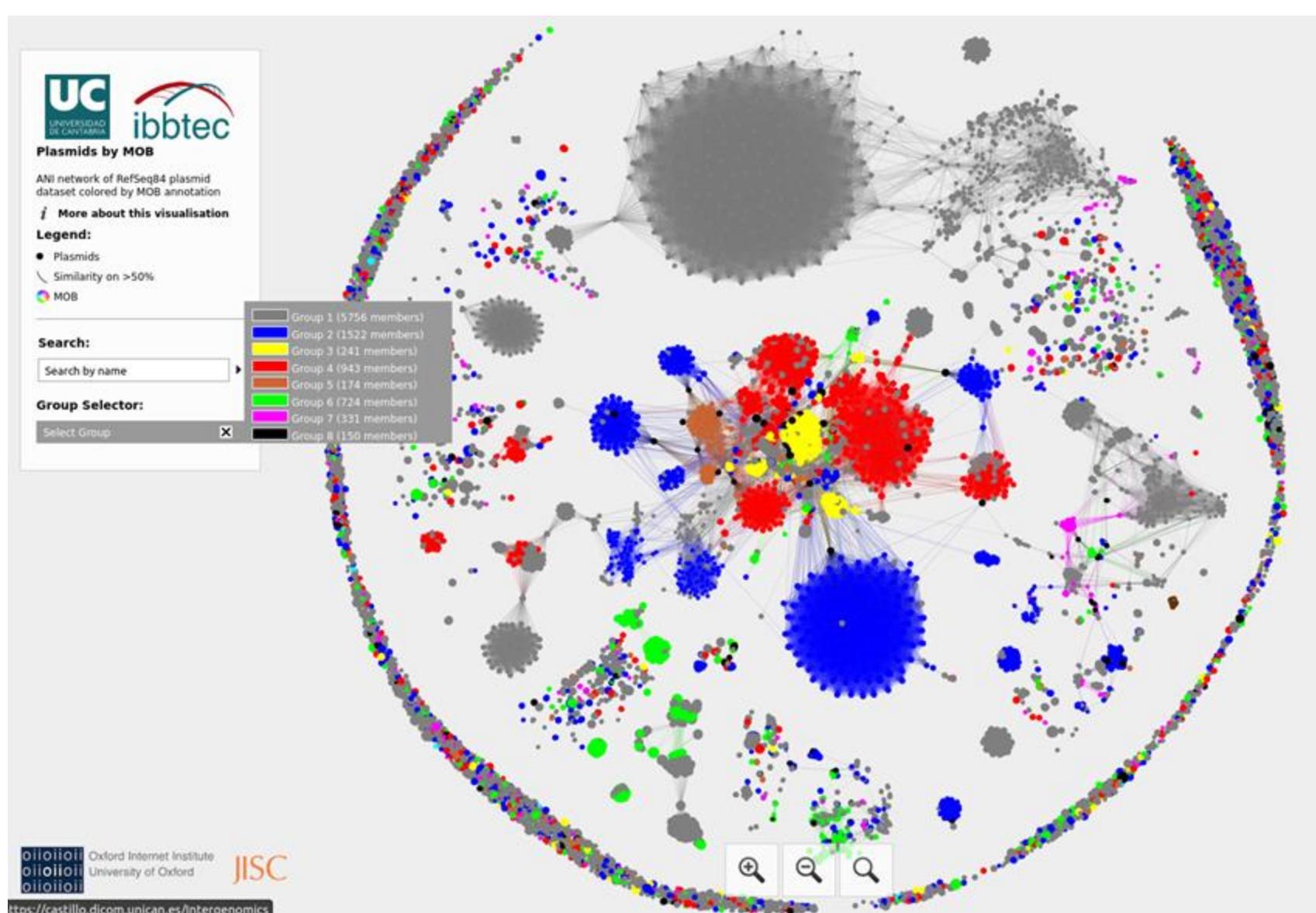
Santiago Redondo-Salvo<sup>1</sup>, Roger Bartomeus-Perñalver<sup>1</sup>, Luis Vielva<sup>2</sup>, Kaitlin A. Tagg<sup>3,4</sup>, Hattie E. Webb<sup>3,4</sup>, Raúl Fernández-López<sup>1</sup> and Fernando de la Cruz<sup>1\*</sup>





List of available networks:

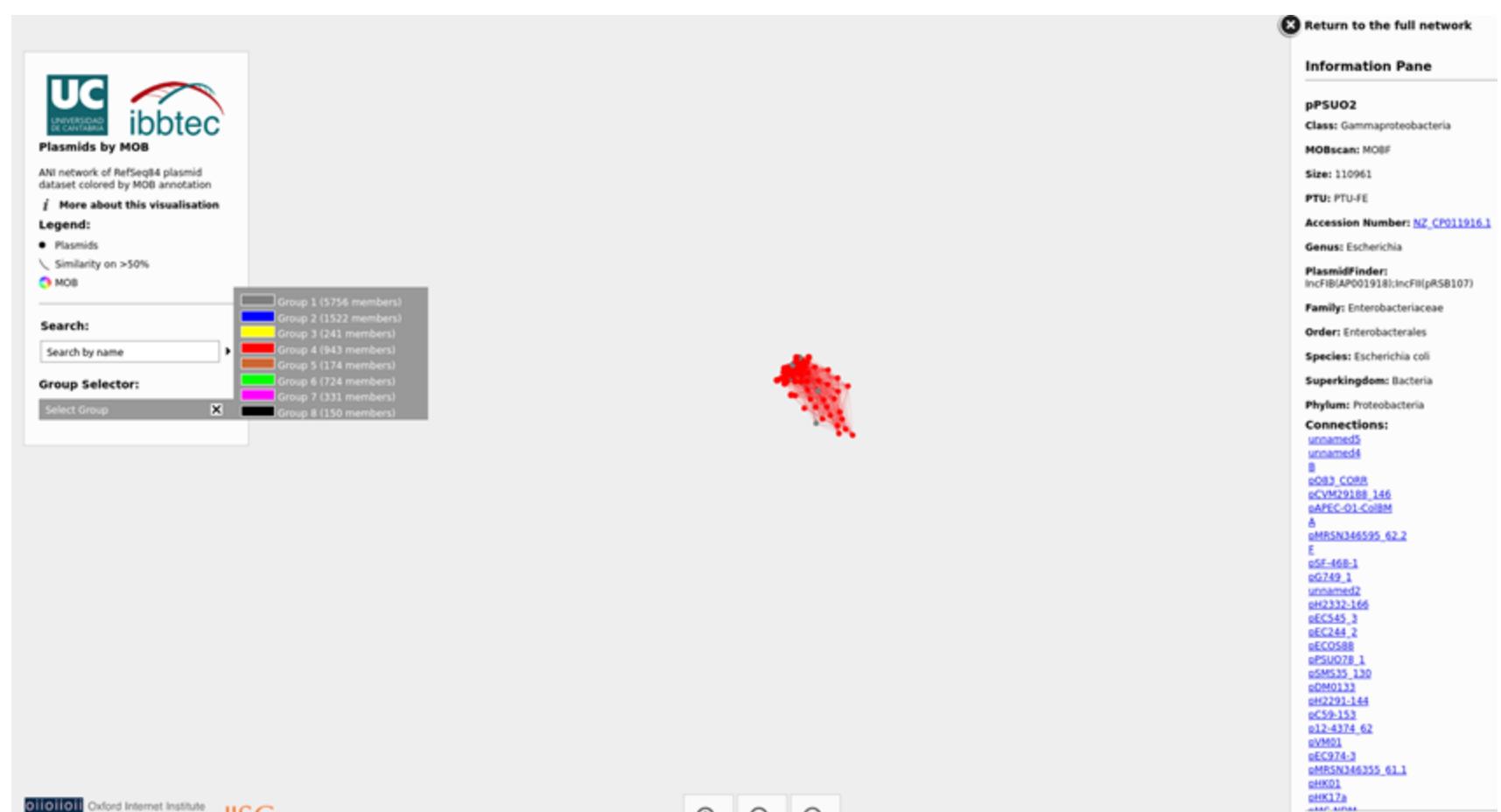
- RefSeq84 dataset: [MOB - Genus](#)
- Enterobacteriales dataset: [MOB - Genus](#)
- host-PTU interactions: [RefSeq84](#)





List of available networks:

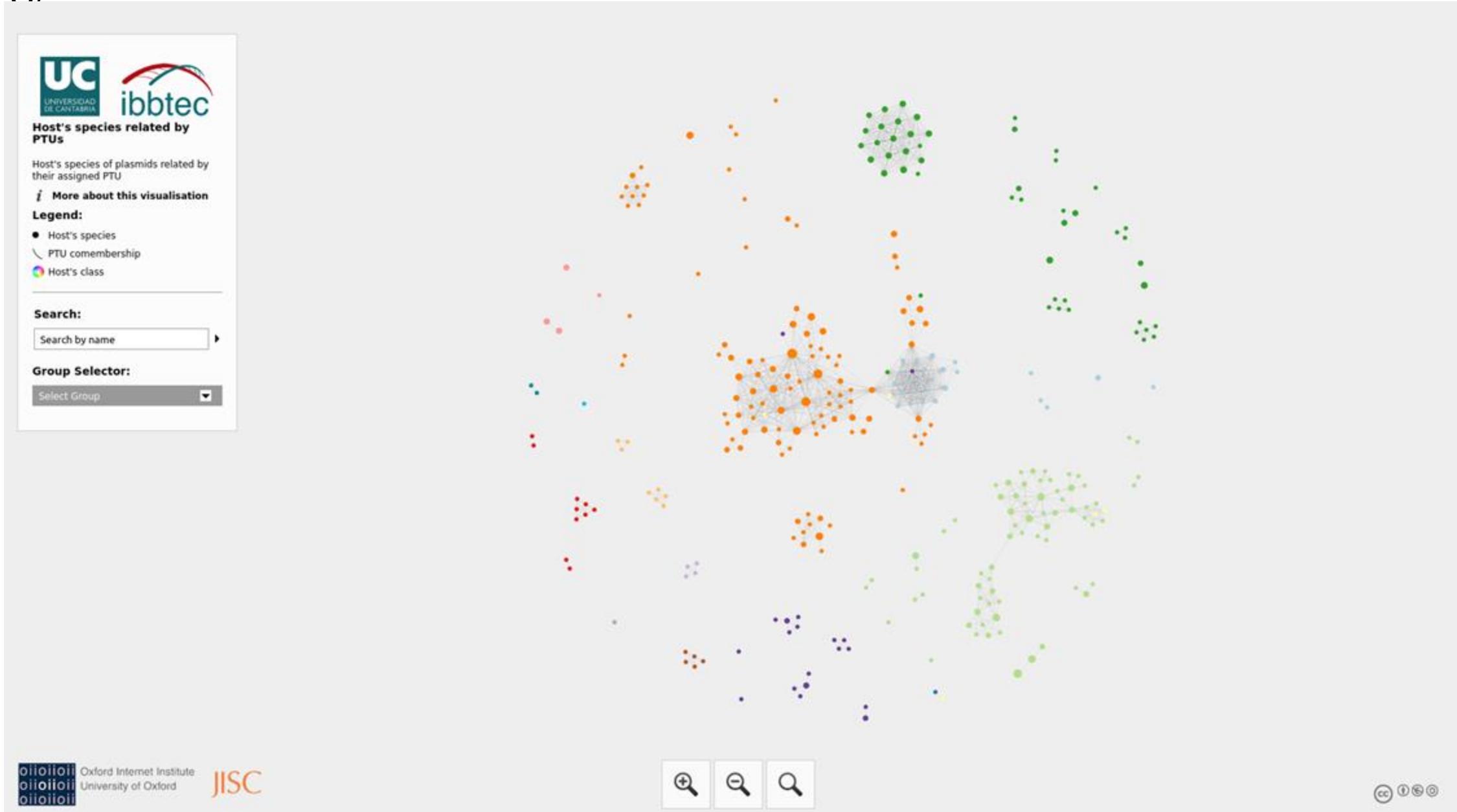
- RefSeq84 dataset: [MOB - Genus](#)
- Enterobacteriales dataset: [MOB - Genus](#)
- host-PTU interactions: [RefSeq84](#)

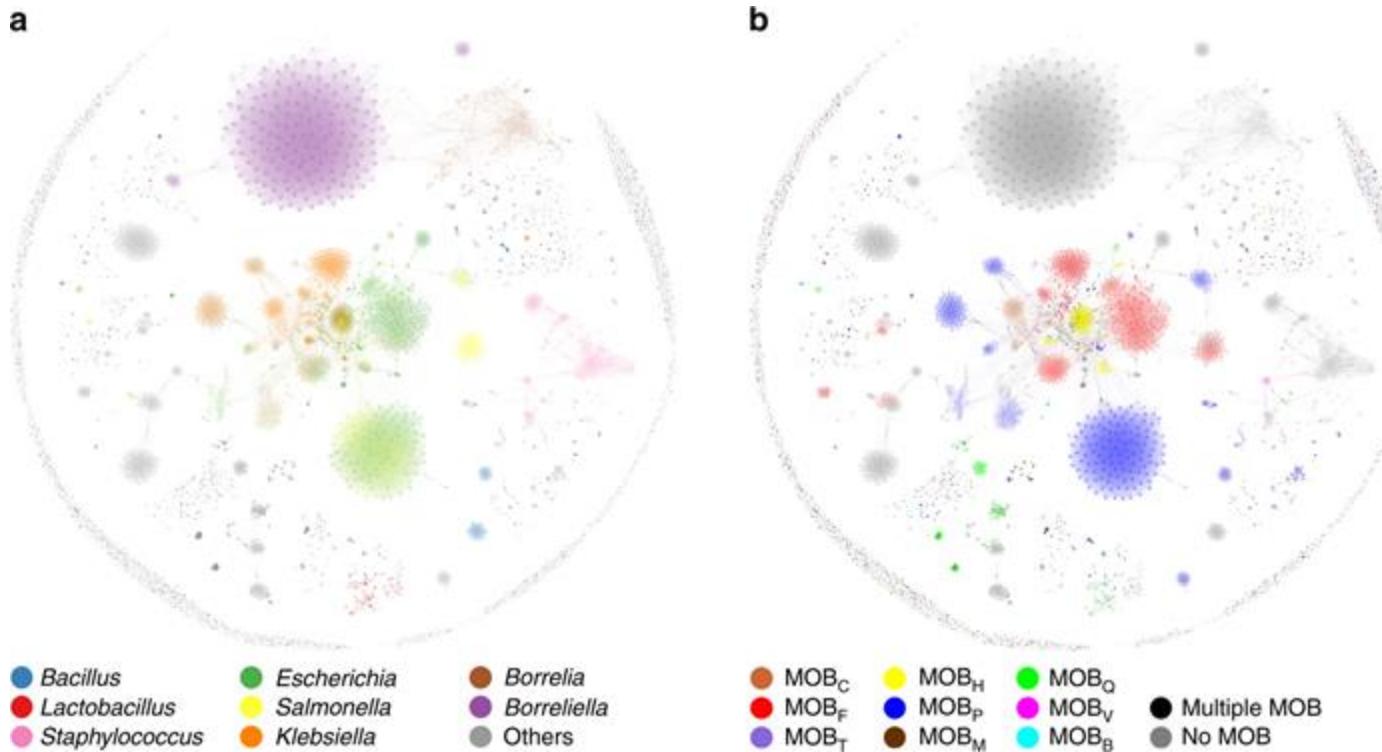




List of available networks:

- RefSeq84 dataset: [MOB](#) - [Genus](#)
  - Enterobacteriales dataset: [MOB](#) - [Genus](#)
  - host-PTU interactions: [RefSeq84](#)

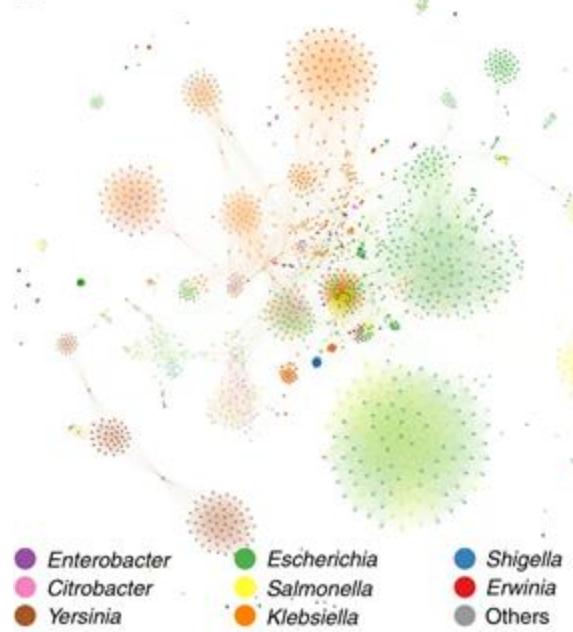
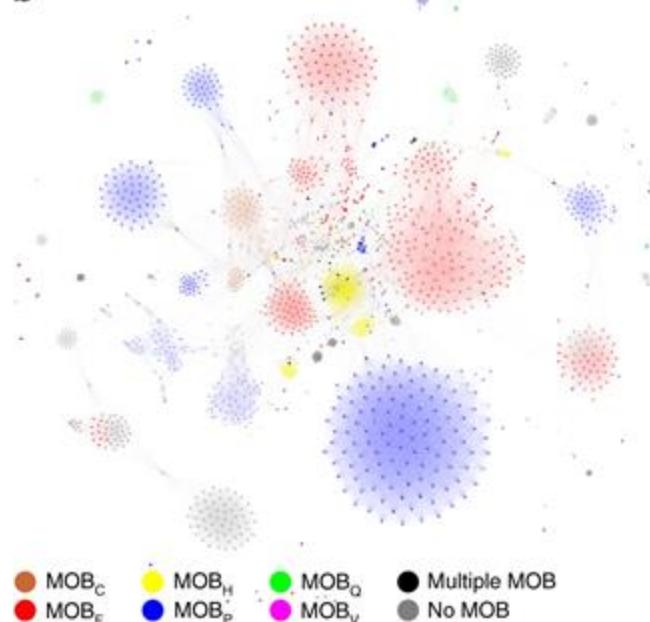




Similarity networks of the RefSeq84 prokaryotic plasmidome obtained using the  $\text{ANI}_{L50}$  algorithm. Nodes, corresponding to plasmid genomes, are colored according to their cognate host taxon or by MOBscan (b). Source data are provided as a Source Data file.

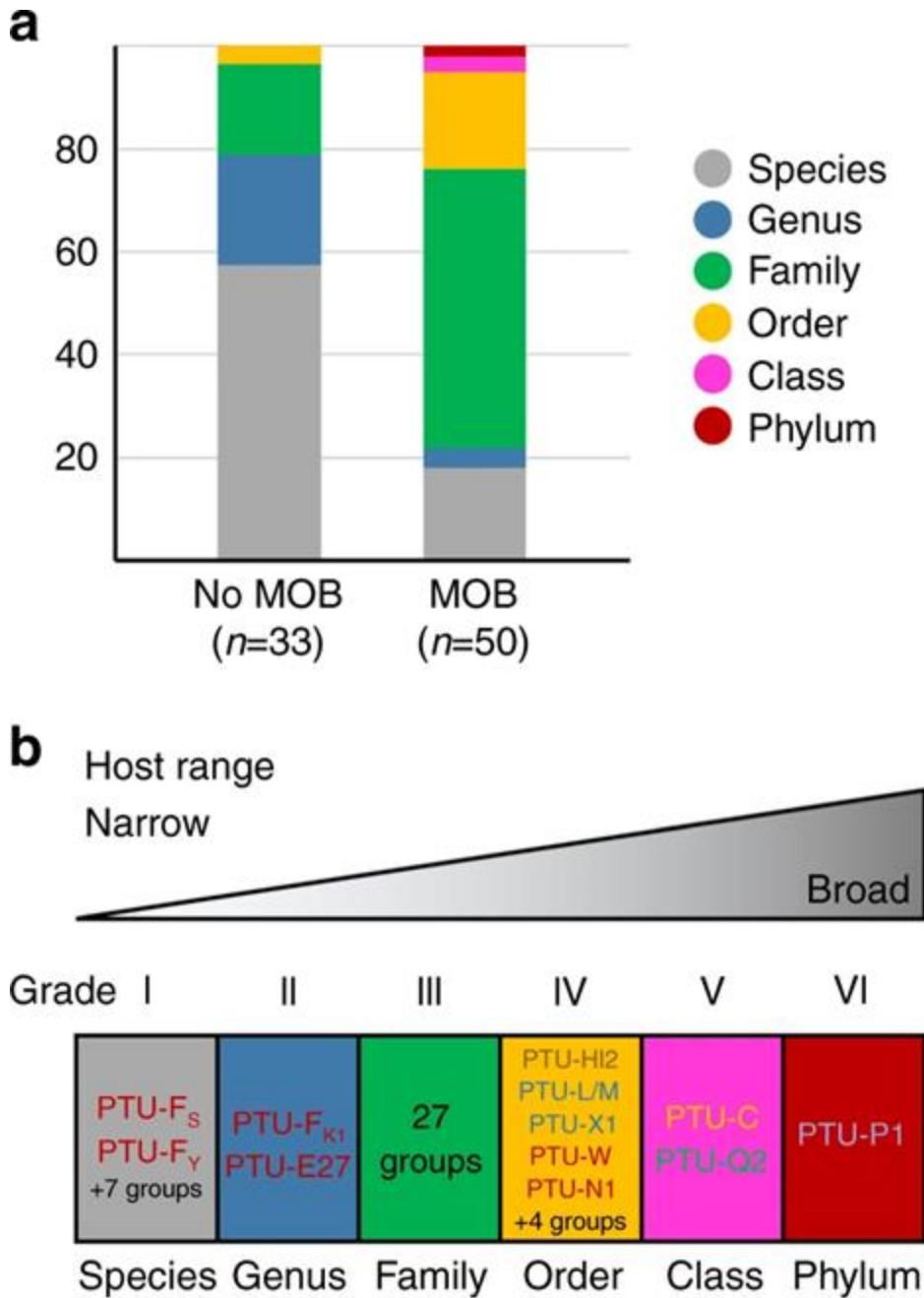
**ANI = Average Nucleotide Identity**

- medida de la similitud a nivel nucleotídico entre dos regiones de un genoma
- actualmente para definición taxonómica de nuevas especies

**a****b****c**

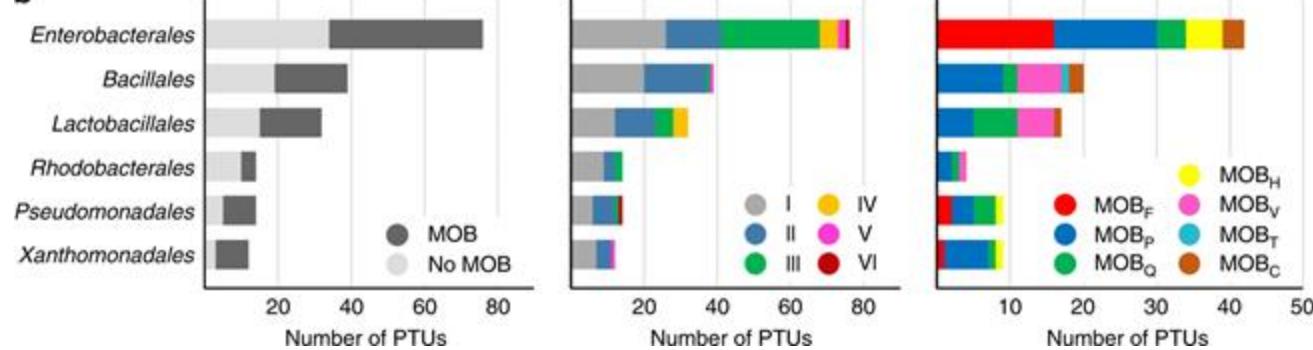
	Eco	Kle	Sal	Shi	Ent	Cit	Ser	Yer	Erw	Pro	Rep. formula	$\delta_C$	Prototype	Size (kb)	
PTU-N1	28	19	4			7				2	0%	50%	100%	91 R46	51
PTU-N2/3	4	2			2	1					IncN	86	p271A	36	
PTU-N3	2	2	1			1				1	N.A.	100	pN-CI	35	
PTU-W	1		2							1	IncW	100	R388	34	
PTU-F <sub>E</sub>	171	12	11	4							IncFil	26	R1 <sup>t</sup> ; F (NC_002483)	98	
PTU-F <sub>K1</sub>			68								IncFilB; IncFil	91	pKPN3	176	
PTU-F <sub>K2</sub>	4	27									IncFilB; IncFil	94	pKpQIL	114	
PTU-F <sub>K3</sub>	2	18									IncFil	62	pKF3-94	94	
PTU-F <sub>S</sub>			56								IncFilB; IncFil	94	pSLT (NC_003277)	94	
PTU-F <sub>Sh</sub>	1					17					IncFil	96	pWR100	213	
PTU-F <sub>V</sub>	2	3								38	IncFil; IncFil	100	pKPC_CAV1099	113	
PTU-F <sub>Y</sub>										6	IncFilB	100	pMT1_(NC_004835)	101	
PTU-E26											pYE854	100	NZ_CP016047	101	
PTU-P1			1								IncP1	57	RP4 <sup>t</sup>	60	
PTU-I1	65	1	49	6						1	IncI1	94	R64	121	
PTU-B/O/K/Z	18			2							IncB/O/K/Z	61	R387 <sup>t</sup> ; pO113	166	
PTU-L/M	4	40				7	3	2		1	IncL/M	99	pCTX-M3	89	
PTU-E25	1	6									IncFil	100	pUCLAQXA232-3	89	
PTU-X1	12		32	4						1	IncX1	75	pOLA52	52	
PTU-X3	6	31			2	2					IncX3	98	pEC14_35	35	
PTU-X4	12	1	6	1						1	IncX4	99	pSAM7	35	
PTU-E1	33		9	8	1						ColRNAl	41	ColE1	7	
PTU-E3	4	16	18		13	6	2			7	N.A.	37	pMdT1	6	
PTU-E14	9		9								ColRNAl	78	pColK-K235	8	
PTU-I2	17	6	6	1							IncI2	97	R721	76	
PTU-E27	4			5							N.A.	100	p14-95A	88	
PTU-E28			2								IncX4	100	pEC4115	37	
PTU-E19	2		5	2		1					Col156	78	pCRO03	4	
PTU-E7	23	1									Col156	71	ColE7-K317	6	
PTU-E10	20		3								Col8282	97	pIGWZ12	4	
PTU-E20	6			3							Col156	94	ColE9-J	8	
PTU-E4	1	37				1	1				ColRNAl	98	ColDF13	10	
PTU-E8		11			9	1	1				repA	100	pKPC_UVA01	44	
PTU-HI1A	2		6								InchI1A; InchI1B	100	R27	180	
PTU-HI1B	1	23									InchI1B; IncFilB	97	pNDM_MAR	267	
PTU-HI2	10	1	11		7	2	1				InchI2; InchI2A; RepA	99	R478	275	
PTU-C	21	32	24							1	IncA/C2	98	pR55	171	

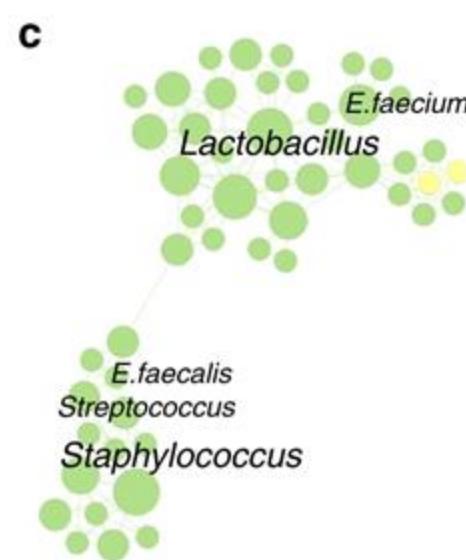
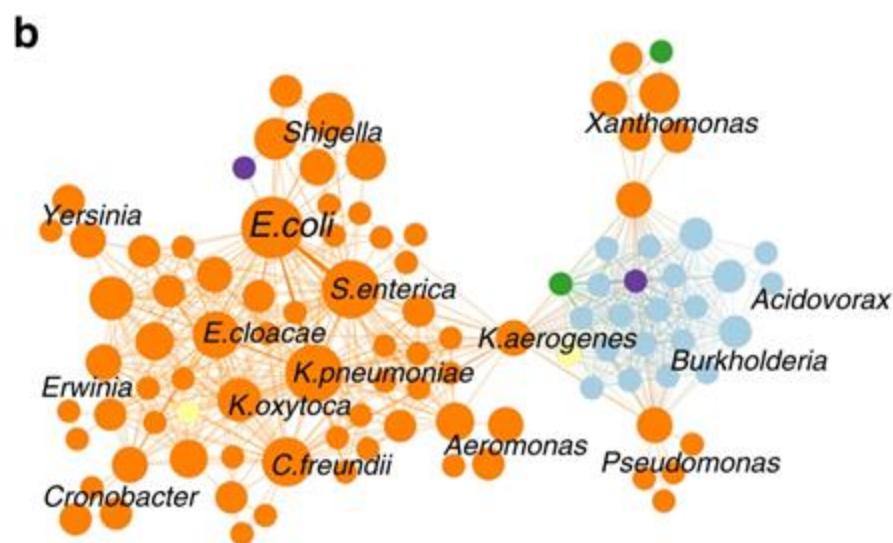
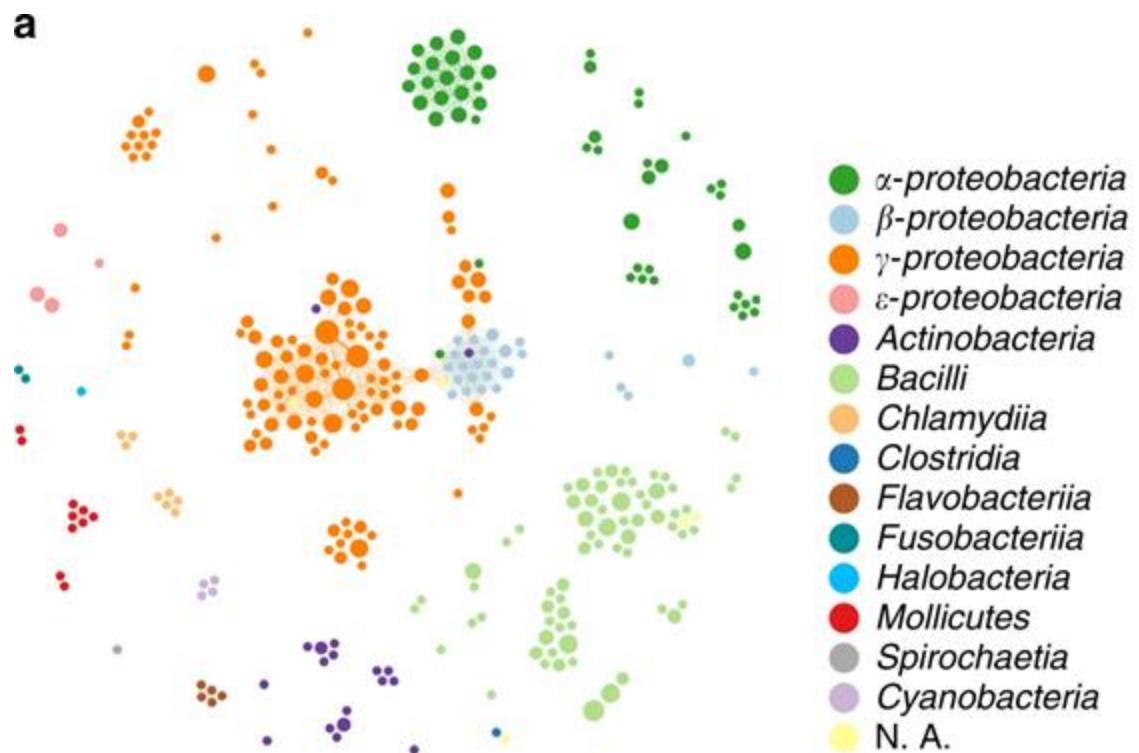
<sup>t</sup>Not in RefSeq84.

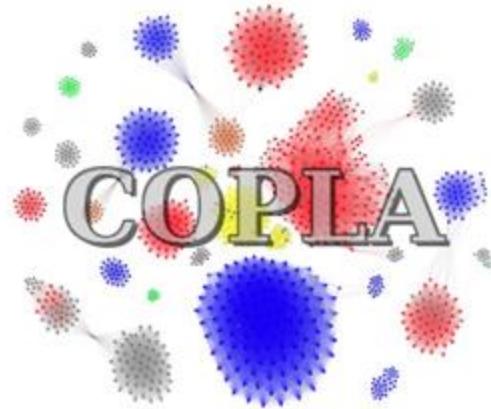


a

b







Please, fill the submission form below and select the FASTA file containing the nucleotide sequence of the plasmid.

**Id of the job\***:

**FASTA nucleotide file (.fna)\***:  No se ha seleccionado ningún archivo.

**FASTA amino acid file (.faa)**:  No se ha seleccionado ningún archivo.

**Contigs topology:**

**Kingdom:**

**Phylum:**

**Class:**

**Order:**

**Family:**

**Genus:**

**Species:**

[Guide](#)

[Source code](#)

[Examples](#)

[About COPLA](#)

santirndn / COPLA Public

Code Issues Pull requests Actions Projects Wiki Security Insights

Go to file Code

main · 1 branch · 0 tags

santrndn Fix too long wildcard expansion · 6660285 on 30 Jun 2021 · 12 commits

bin · Fix too long wildcard expansion · 12 months ago

test · Fix test output files · 13 months ago

.gitattributes · Initial commit · 2 years ago

LICENSE · Initial commit · 2 years ago

Outcomes.png · Update README.md · 2 years ago

README.md · Fix bug in README.md · 14 months ago

RS200\_plasmid\_metadata.tsv · Upload input metadata · 2 years ago

RS84\_plasmid\_metadata.tsv · Upload input metadata · 2 years ago

Results.png · Update README.md · 2 years ago

copla.environment.yml · Upload code and conda environments · 2 years ago

copla.ini · Upload code and conda environments · 2 years ago

copla.spec-file.txt · Upload code and conda environments · 2 years ago

macsyfinder.environment.yml · Aesthetic fix · 13 months ago

macsyfinder.spec-file.txt · Update MacSyllinder environment · 14 months ago

README.md

**COPLA, a taxonomic classifier of plasmids**

About

A taxonomic classifier of plasmids

[castillo.dicom.unican.es/copla/](#)

Readme

GPL-3.0 license

14 stars

2 watching

3 forks

Releases

No releases published

Packages

No packages published

Languages

Python 78.1% Shell 18.9% Perl 3.0%

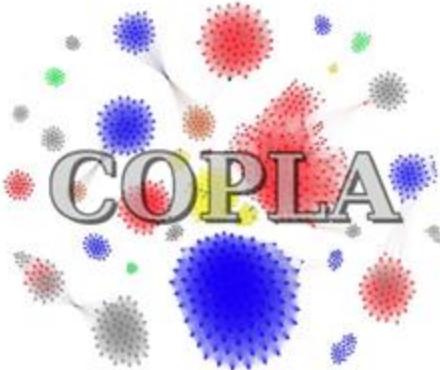
## How COPLA works

Just as sHSBM, COPLA infers the PTU membership from the similarity relationships between plasmids in the database. After calculating the ANI percentage identity between the query and the reference plasmid set, COPLA inserts the query into the reference network and performs a statistical search for similar plasmids. Finally, the query is assigned to a known PTU, or to a new PTU (labeled as PTU-?) if the algorithm finds clues pointing to that outcome. PTUs will not be named for clusters with fewer than 4 members. For the user to evaluate the COPLA PTU assignment of a query plasmid, a score is provided based on the overlap of the graph partitions before and after the query was inserted into the reference network (for additional details see troubleshooting below).

## COPLA output

COPLA output consists of five different files:

- **query.ptu\_prediction.tsv**: File providing the predicted PTU of the query, the graded PTU host range (updated by the taxonomic info of the query if provided by the user), the prediction score, and additional information (see troubleshooting) to help interpret the output.
- **query.related\_plasmids.tsv**: List of reference plasmids used for the calculation of the prediction score.
- **query.ani.tsv**: List of ANI percentage identities between the query and reference plasmids. Only plasmids with a detected homology spanning >50% of the shorter genome are reported (Best reciprocal BLASTn hits of 1,000 bp genome fragments with >70% identity over >70% length. Fragments are obtained using a sliding window algorithm with 200 bp steps). File columns indicate query, reference accession number, ANI, standard deviation of ANI, fragments used for the calculation of ANI and fragments of the reference genome.
- **query.qry\_info.tsv**: Other relevant plasmid information: genome length, MOB typing (according to MOBscan), MPF typing (according to CONJScan but with the relaxase HMM profiles updated to those of MOBscan database), replicon typing (based on PlasmidFinder using 80% identity threshold) and AMR genes (based on a BLASTn search against the CARD database)
- **query.faa**: If not provided by the user, this file is the plasmid ORFeome calculated by Prodigal. Prodigal autolearning mode is used for plasmids >100 Kb, smaller plasmids are calculated using Prodigal meta mode.

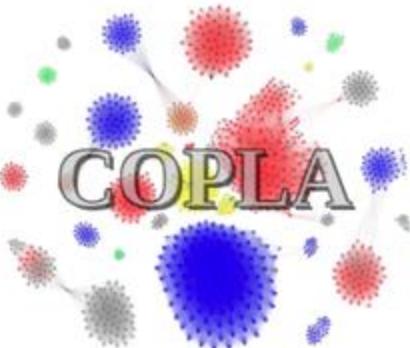


Please, fill the submission form below and select the FASTA file containing the nucleotide sequence of the plasmid.

Id of the job*:	<input type="text" value="FC2"/>	<input type="button" value="Upload"/>
FASTA nucleotide file (.fna)*:	<input type="button" value="Examinar..."/> FC2.fasta	
FASTA amino acid file (.faa):	<input type="button" value="Examinar..."/> No se ha seleccionado ningún archivo.	
Contigs topology:	<input type="button" value="linear"/>	
Kingdom:	<input type="text" value="Bacteria"/>	
Phylum:	<input type="text" value="Proteobacteria"/>	
Class:	<input type="text" value="Gammaproteobacteria"/>	
Order:	<input type="text" value="Enterobacterales"/>	
Family:	<input type="text" value="Enterobacteriaceae"/>	
Genus:	<input type="text" value="Escherichia"/>	
Species:	<input type="text" value="Escherichia coli"/>	
<input type="button" value="Upload"/>		

Your data is being processed. The web page will update automatically when finished. Please be patient as it could take a few minutes.

100%



Please, fill the submission form below and select the FASTA file containing the nucleotide sequence of the plasmid.

Id of the job*:	<input type="text" value="Enter your job name"/>
FASTA nucleotide file (.fna)*:	<input type="button" value="Seleccionar archivo"/> Ningún archivo seleccionado
FASTA amino acid file (.faa):	<input type="button" value="Seleccionar archivo"/> Ningún archivo seleccionado
Contigs topology:	<input type="button" value="linear"/>
Kingdom:	<input type="text" value="Bacteria"/>
Phylum:	<input type="text" value="Proteobacteria"/>
Class:	<input type="text" value="Gammaproteobacteria"/>
Order:	<input type="text" value="Enterobacterales"/>
Family:	<input type="text" value="Enterobacteriaceae"/>
Genus:	<input type="text" value="Escherichia"/>
Species:	<input type="text" value="Escherichia coli"/>
<input type="button" value="Upload"/>	

Your data has been successfully processed. The complete report can be downloaded from [here](#). Please copy and save it since it will remain accessible only for one week.

#Predicted_PTU	Host_Range	Score	Notes
PTU-FE	III	1.0000	Query is a PTU-FE plasmid

Nombre
query.fna_amr
query.fna_conjscan
query.fna_mobscan
query.fna_pfinder
log.txt
query.fna
query.fna.ani.tsv
query.fna.faa
query.fna.ptu_prediction.tsv
query.fna.qry_info.tsv
query.fna.related_plasmids.tsv

## query.ptu\_prediction.tsv

FC2

#Predicted_PTU	Host_Range	Score	Notes
PTU-FE	III	1.0000	Query is a PTU-FE plasmid
<b>query.related_plasmids.tsv</b>			

## **query.related plasmids.tsv**

A	B	C	
#Accession	Version	PTU_Ref	Note
NC_005327.1		PTU-FE	*
NC_006671.1		PTU-FE	*
NC_007675.1		PTU-FE	*
NC_010409.1		PTU-FE	*
NC_011812.1		PTU-FE	*
NC_011964.1		PTU-FE	*
NC_013175.1		PTU-FE	*
NC_013122.1		PTU-FE	*
NC_013542.1		PTU-FE	*
NC_014382.1		PTU-FE	*
NC_014615.1		PTU-FE	*
NC_013121.1		PTU-FE	*
NC_016039.1		PTU-FE	*
NC_018998.1		PTU-FE	*
NC_019037.1		PTU-FE	*
NC_019071.1		PTU-FE	*
NC_019072.1		PTU-FE	*
NC_019073.1		PTU-FE	*
NC_019095.1		PTU-FE	*

# query.qry\_info.tsv

# query.ani.tsv

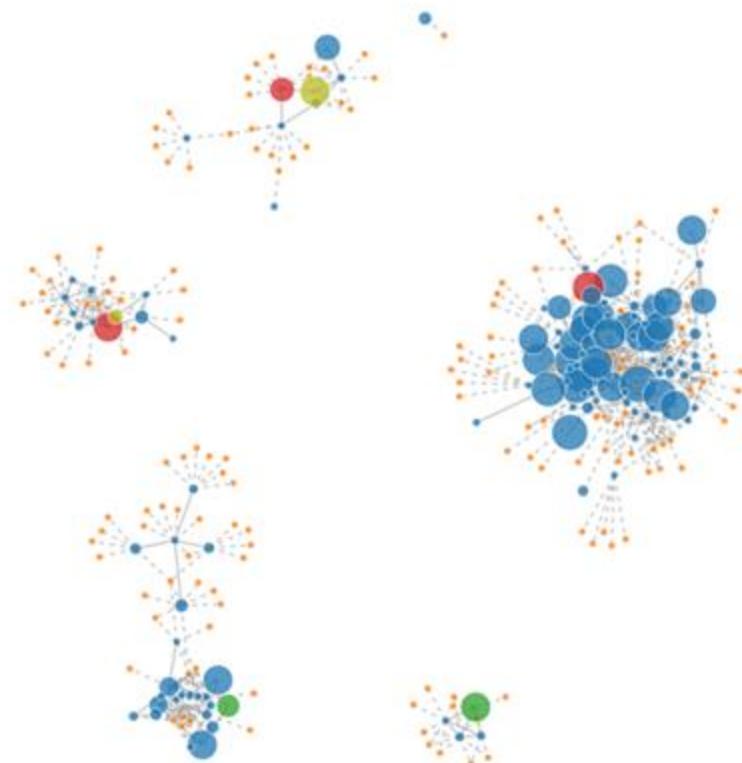
L53

	Predeterminado	Predeterminado	Predeterminado	Predeterminado
1	#Predicted_PTU	Host_Range	Score	Notes
2	-	-	1.0000	PTU could not be assigned. Query is part of a sHSBM cluster of size 1

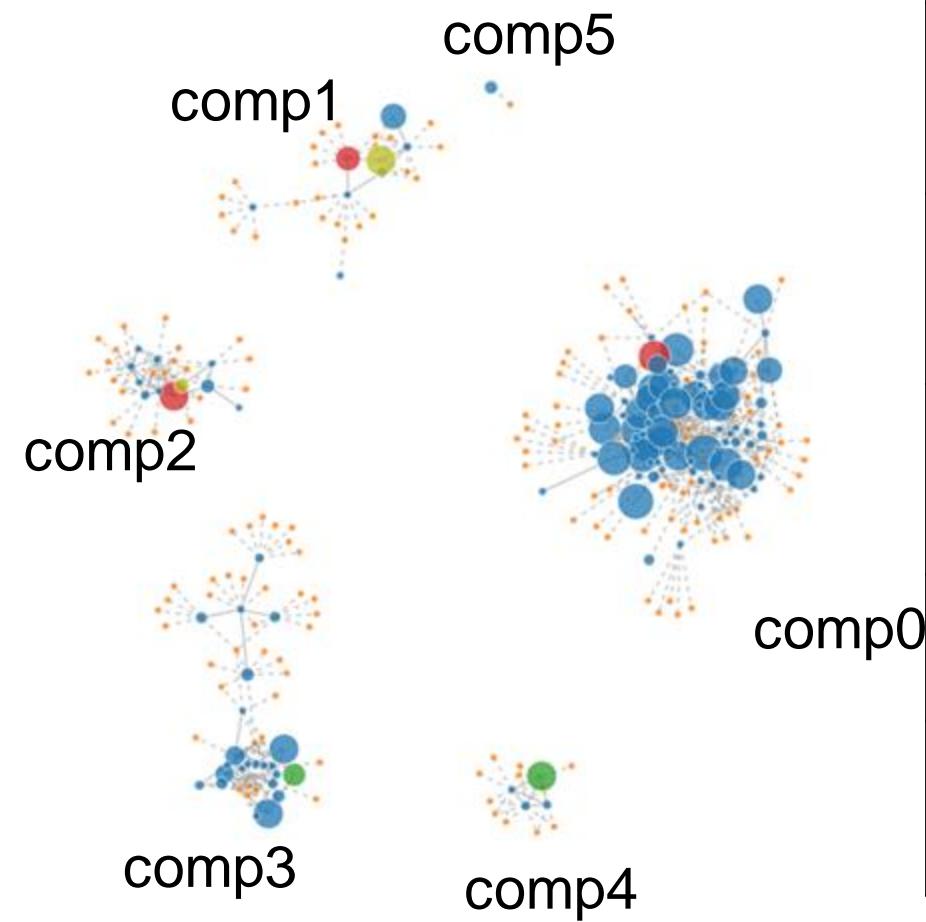
A	B	C	D	E
#Total_bp	MOB	MPF	Replicon	AMR
5177774	MOBF;MOBF;MOBH;MOBP;MOBP	typeF:typeF:typeT	Col156;IncFIB(AP001918);IncFI(29);IncL/M(pOXA-48);IncN;IncX1	CRP;Escherichia;Escherichia;Escherichia;Escherichia;Escherichia;Escherichia;Escherichia;H-NS;OXA-48;OmpA;TEM-181;acrB;acrD;acrE;acrF;acrS;bacA;baeR;baeS;cpxA;emrA;emrB;emrK;emrR;emrY;epmA;evgA;evgS;gadW;gadX;kdpE;marA;mdtA;mdtB;mdtC;mdtE;mdtF;mdtG;mdtH;mdtM;mdtN;mdtO;mdtP;mphA;msbA;pmrF;tet(B);tolC;ugd;yoil

Nota: no es aconsejable  
incorporar todo el genoma a la  
vez!

elementos genéticos por  
separado!



L53



comp	#contigs	size	REL/RIP/Inc	PTU	Host range
comp0	79	4.9 Mb	MOBH	PTU-?*	III
comp1	8	62.4 Kb	MOBP RptZ IncP	PTU-L/M	IV
comp2	11	41.3 Kb	MOBF RptF IncN	PTU-N1	IV
comp3	24	130.6 Kb	MOBF RptA1 IncFrepb	PTU-FE	III
comp4	4	34.5 Kb	MOBP RptF	PTU-E80	IV
comp5	1	5.5 Kb**	-	PTU could not be assigned. Query is part of a graph component of size 1	

\*New (putative) PTU. Query is part of a sHSBM cluster of size 6

\*\*pS1a = PhiX internal Illumina control

NUEVO!!!!

Diferencias entre el  
ensamblaje de lecturas  
cortas y largas -  
EXAMEN

5.8.

Ensamblaje de lecturas largas Oxford  
Nanopore

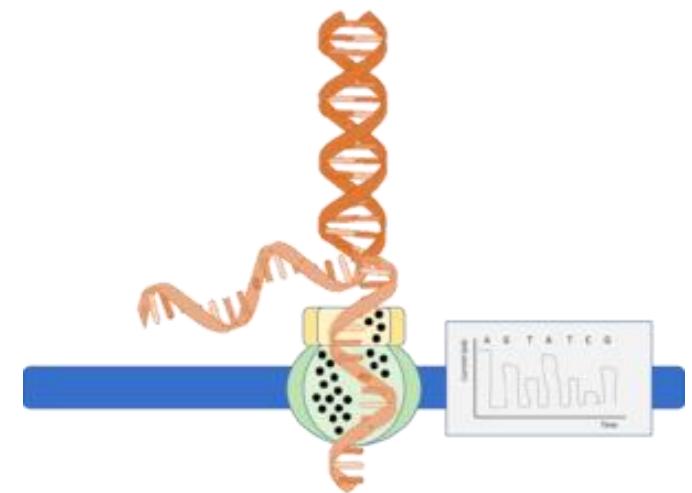
## Secuenciación en Nanoporos

Tras la aparición de Illumina, surgió otra estrategia que ha entrado en el mercado como una auténtica revolución por su amplia versatilidad. Es el caso de la técnica de secuenciación mediante **Nanoporos**, un método de secuenciación que permite secuenciar **fragmentos de ADN de manera individual, de forma directa, barata y a tiempo real**.

Aunque se encuentra continuamente en optimización (sept 2022 lanzaron la química Q20+ para aumentar sensibilidad), este equipo es capaz de secuenciar toda la longitud de la secuencia a analizar, en lugar de generar lecturas de un tamaño específico. A esto es lo que se comienza a llamar como capacidad de ofrecer **lecturas “ultra largas”**.

Todo esto se consigue, gracias a que esta tecnología está basada en el principio de leer directamente el fragmento de ADN a secuenciar mediante el paso de dicha molécula única por un nanoporo, a la vez que se va midiendo los efectos, a nivel iónico y eléctrico, que supone el avance del ADN.

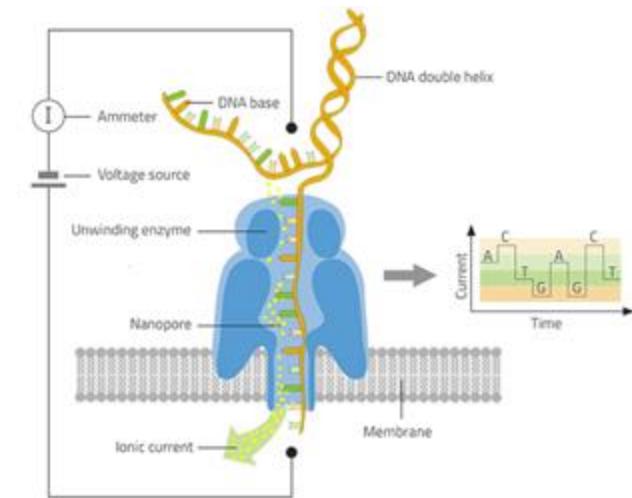
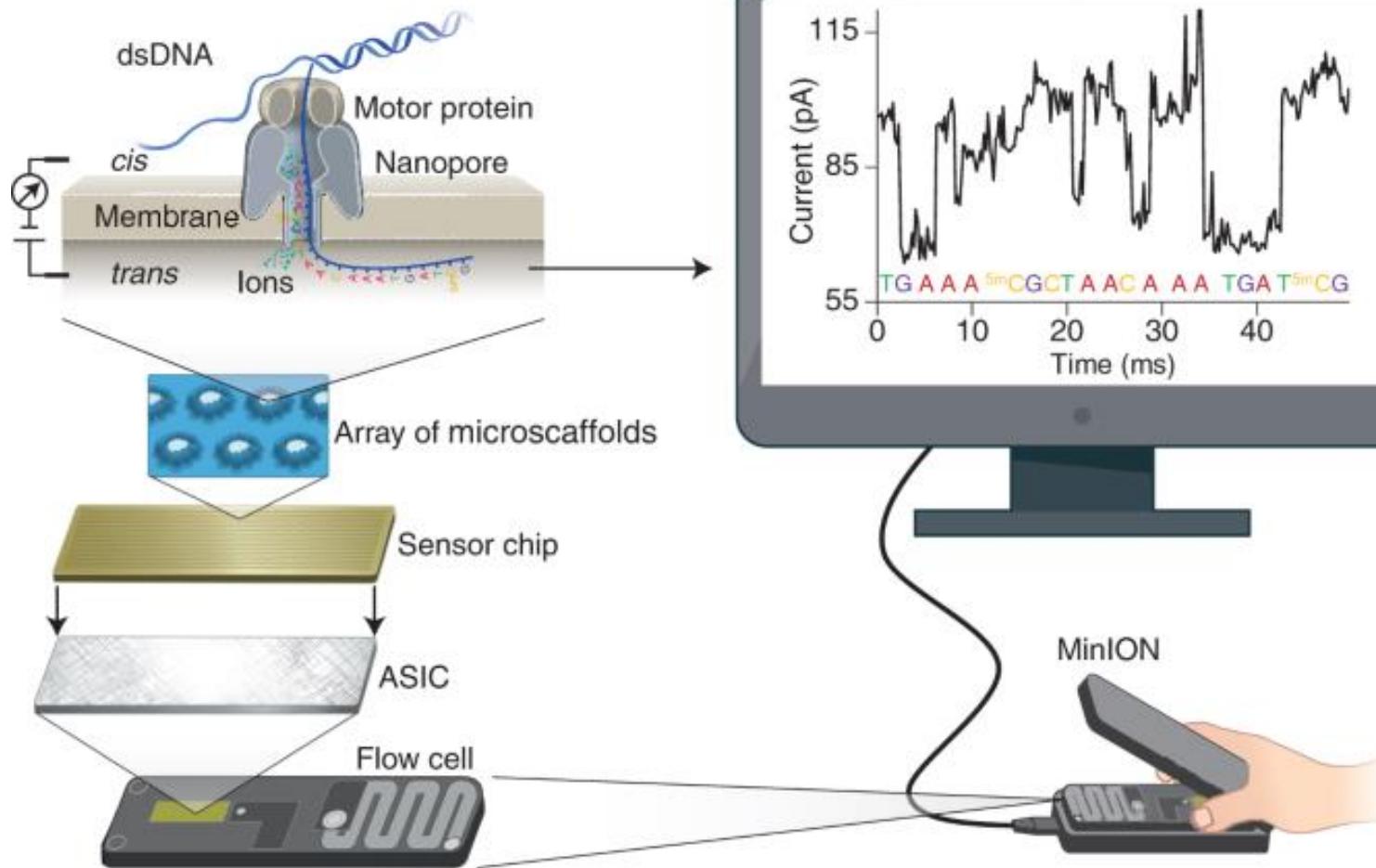
Al igual que en SMRT, la secuenciación mediante nanoporos también presenta la limitación de que se requiere de una **gran cantidad de ADN**, para poder secuenciar y obtener un resultado fiable. Secuenciar una única molécula, tiene el inconveniente, es que se necesita de la secuenciación de muchas moléculas únicas para poder obtener una secuencia consenso aceptable entre todas las idénticas y se eliminan los posibles errores introducidos.

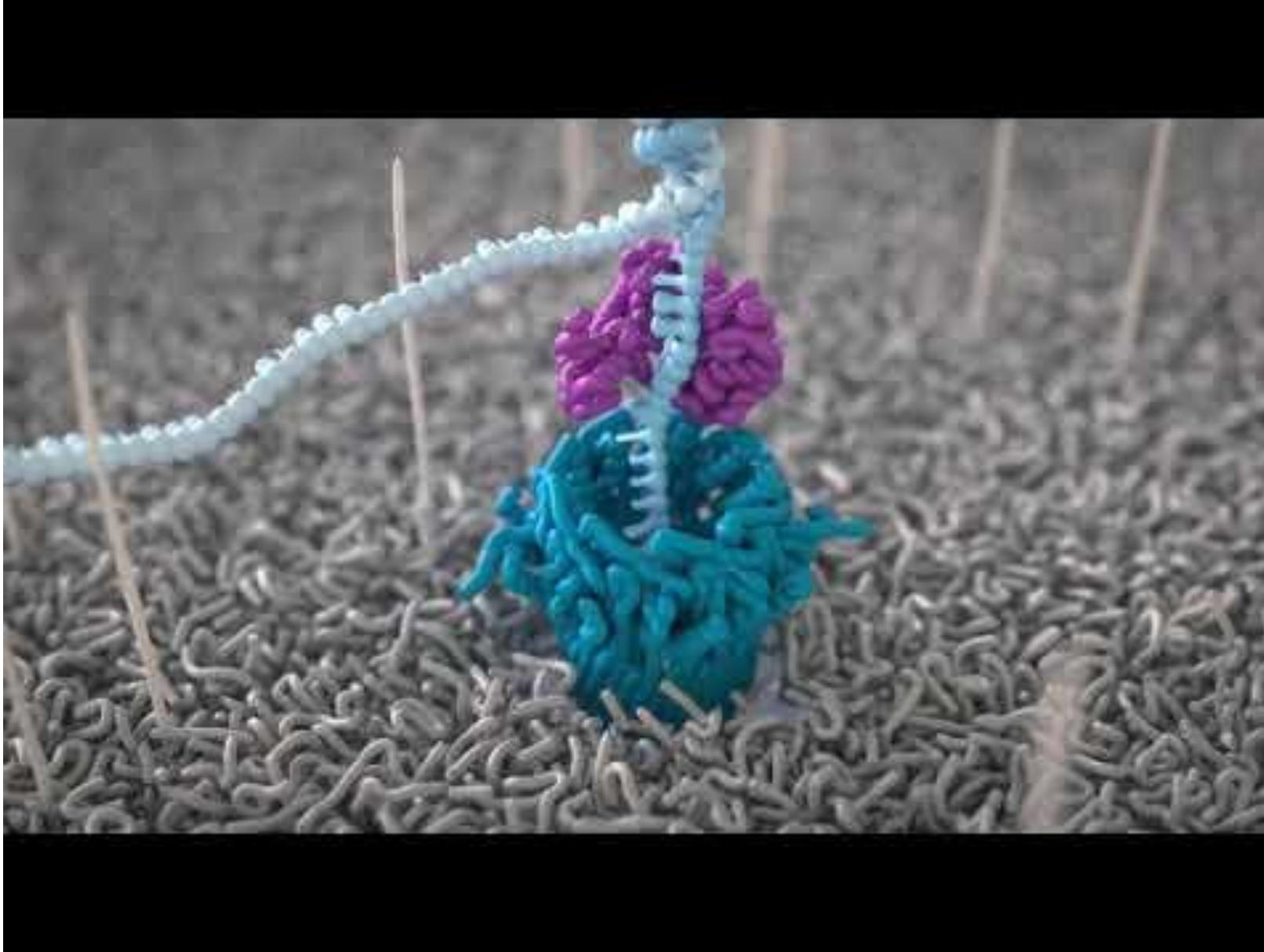


## Nanopore sequencing technology, bioinformatics and applications

Yunhao Wang, Yue Zhao, Audrey Bollas, Yuru Wang & Kin Fai Au 

*Nature Biotechnology* 39, 1348–1365 (2021) | [Cite this article](#)



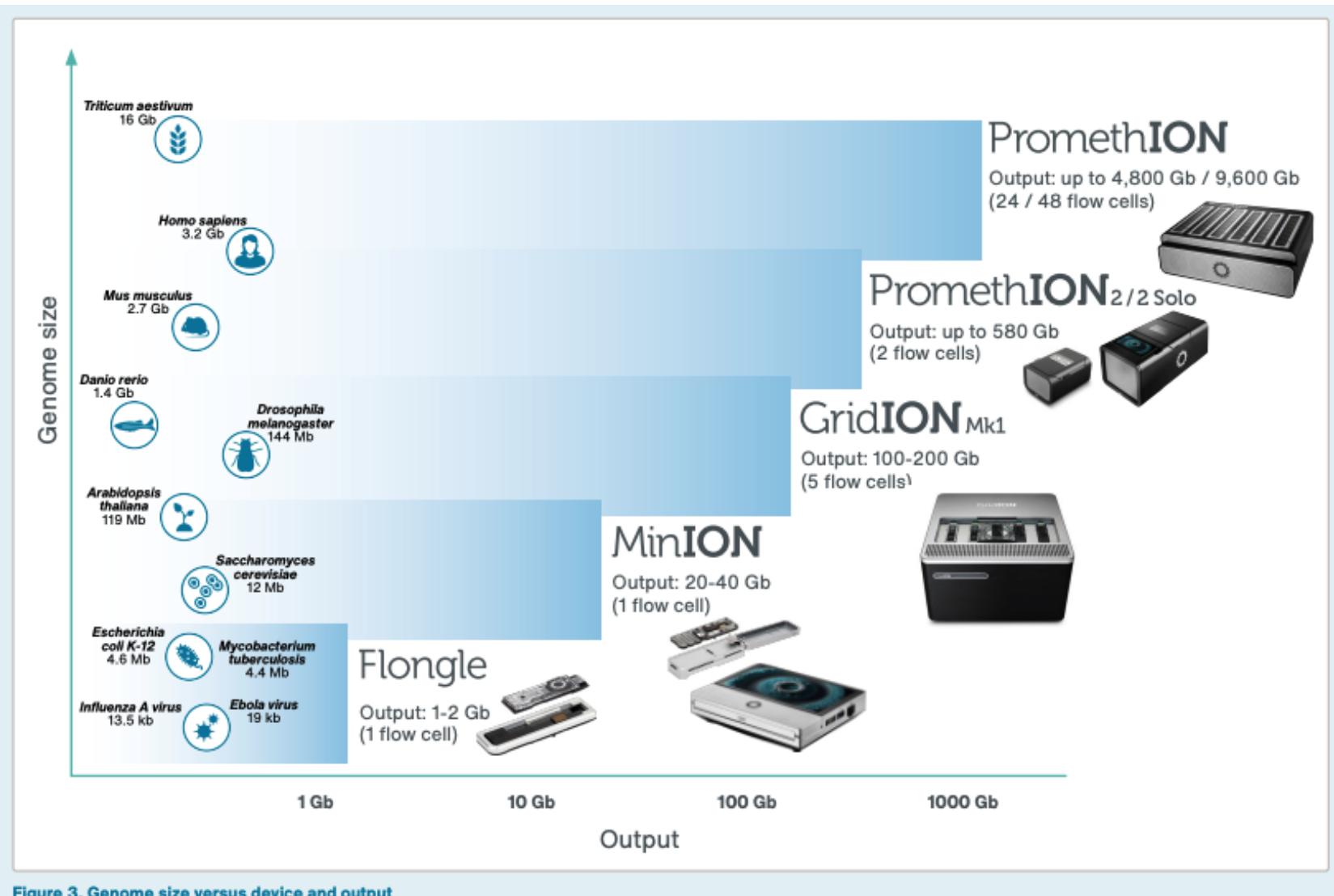


# NANOPORE PRODUCT FAMILY

One core technology, real-time, on-demand



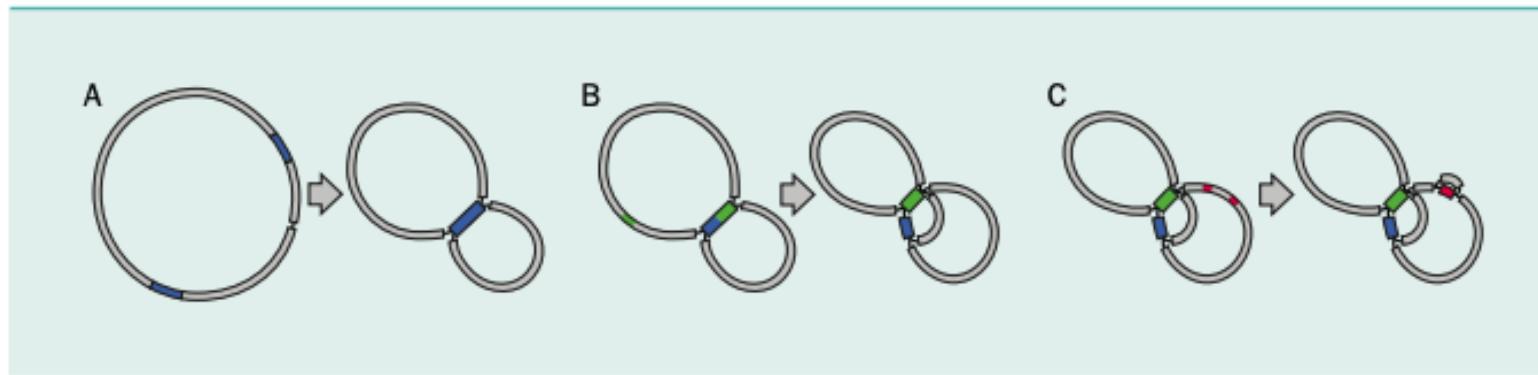
Key	SmidgION	Flongle	MinION	GridION	PromethION
System Price	TBC	Included in \$5K Starter Pack	Included in \$1K Starter Pack	Included in \$50K Starter Pack	Included in \$135K Starter Pack
Number of channels	200 channels	128 channels	512 channels	$5 \times 512 = 2,560^*$	$48 \times 3,000^* = 144,000$
Per flow cell Current Data – Max Data	TBC	1 - 3.3 Gb	17 - 40 Gb	17 - 40 Gb	125 - 311 Gb
Per Device Current Data – Max Data				85 - 200 Gb	3/6 - 20 Tb
Price per Gb Current Data – Max Data	<b>TBC</b>	<b>\$90 - \$30</b>	<b>\$30 - \$12.5</b>	<b>\$17.5 - \$7.5</b>	<b>\$5 - \$2</b>



	<b><i>Second-generation sequencing</i></b>	<b><i>Third-generation sequencing</i></b>
<b>Ventajas</b>	<ul style="list-style-type: none"> <li>Alta precisión de secuencia</li> <li>Bajo coste económico</li> <li>Posibilidad de secuenciar ADN fragmentado</li> </ul>	<ul style="list-style-type: none"> <li>Posibilidad de empezar con fragmentos grandes de DNA</li> <li>Fácil preparación de librería (portable) *</li> <li>Epigenética: patrones de metilación y modificaciones de histona</li> <li>Genera secuencias largas, muy largas</li> </ul>
<b>Desventajas</b>	<ul style="list-style-type: none"> <li>Solo produce secuencias cortas (hasta 300bp PE)*</li> <li>No resuelve variantes estructurales</li> <li>No distingue regiones homólogas del genoma</li> <li>Problemas en las regiones altamente repetitivas</li> <li>No detecta epigenética</li> </ul>	<ul style="list-style-type: none"> <li>Baja precisión de secuencia</li> <li>Elevado coste económico</li> </ul>

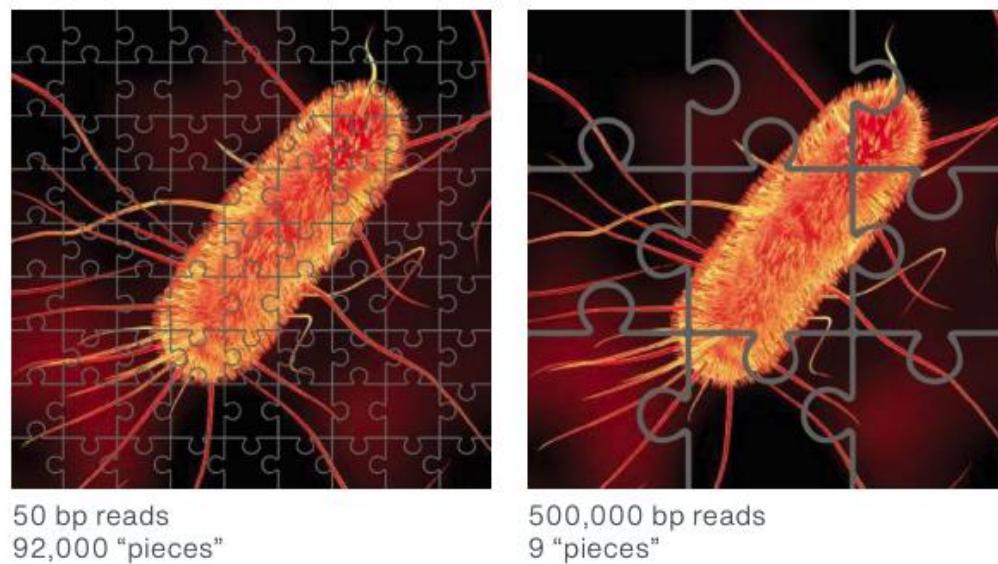
## ¿Qué tecnología escoger para mi proyecto?

<https://frontlinegenomics.com/dna-sequencing-how-to-choose-the-right-technology/>



**Figure 2**

Schematic highlighting the challenge of correct genome assembly using short-read data. The higher the number of repeats (blue, green, and red regions), the more tangled the assembly graphs become, resulting in fragmentation of the assembly. Image courtesy of Ryan Wick, University of Melbourne, Australia.



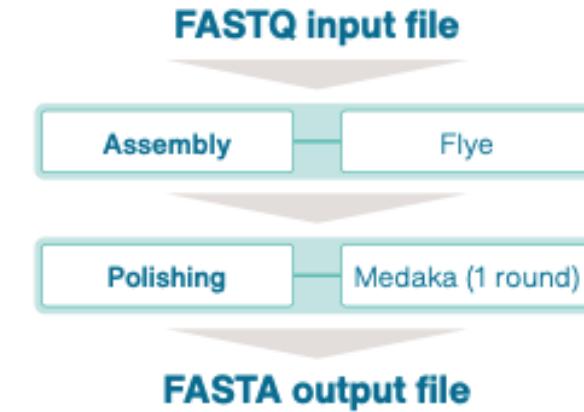
## ANALYSIS: selecting an assembly tool

To perform bacterial genome assembly, we suggest using the third-party *de novo* assembly tool **Flye**<sup>3</sup>. This analysis package represents a complete pipeline, taking raw nanopore reads as input, and producing polished contigs as output. We also recommend one round of polishing with **Medaka**<sup>4</sup>. These tools can be found on GitHub.

Bacterial genomes up to ~10 Mb can be assembled with Flye using a standard laptop (~16 GB memory).

This complete analysis pipeline is also available in EPI2ME Labs, which provides best practice workflows and interactive tutorials to support the analysis of your nanopore sequencing data and develop your bioinformatics skills. Find out more at [labs.epi2me.io](https://labs.epi2me.io).

Find out more about data analysis solutions:  
[nanoporetech.com/analyse](http://nanoporetech.com/analyse)

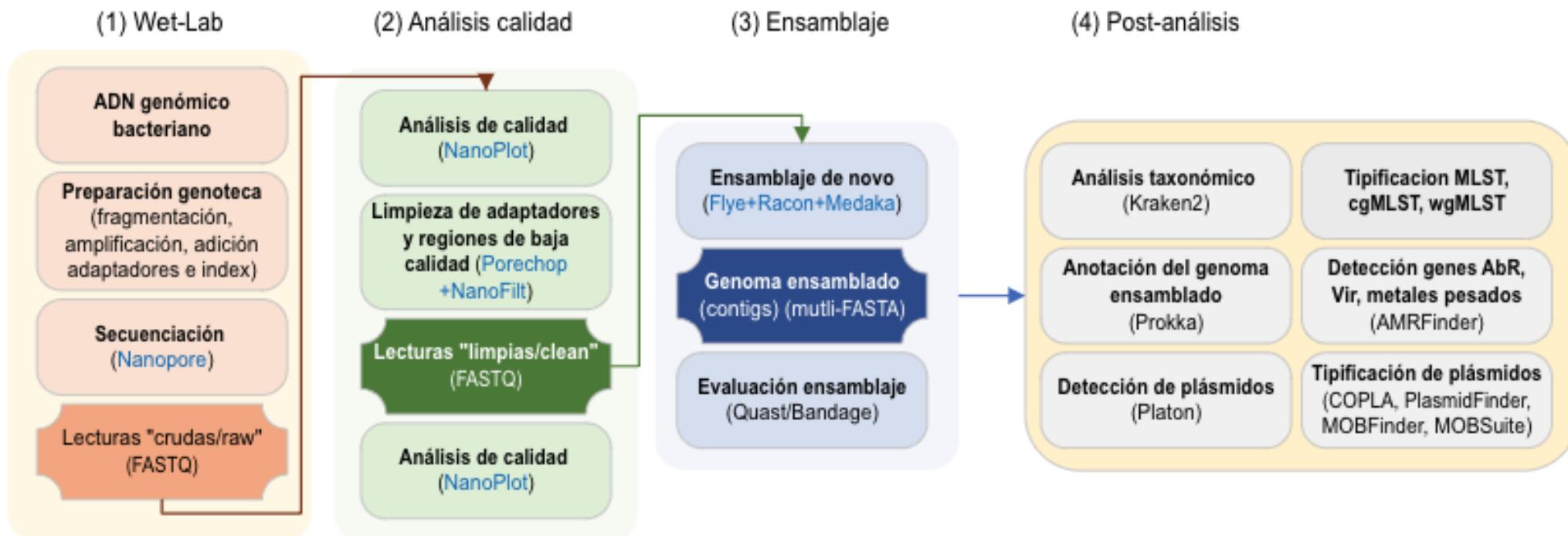


<https://github.com/rrwick/Trycycler/wiki/Guide-to-bacterial-genome-assembly>

<https://github.com/rrwick>

# Environment para datos Nanopore

Instalación Mamba:					
Environment*	Programa	Comando descarga	Versión	Utilidad	Anotaciones (extras a instalar, referencias...)
04MBIF_bacteriano_ONT		mamba create -n 04MBIF_bacteriano_ONT python=3.8 mamba activate 04MBIF_bacteriano_ONT			
	nanoQC	mamba install -c bioconda nanoqc	v0.9.4	análisis de FastQ de ONT	<a href="https://github.com/wdecoster/nanoQC">https://github.com/wdecoster/nanoQC</a>
	NanoStat	mamba install -c bioconda nanostat	v1.6.0	ploteo de calidad de secuencias (compatible con MultiQC)	<a href="https://github.com/wdecoster/nanostat">https://github.com/wdecoster/nanostat</a>
	MultiQC	mamba install -c bioconda multiqc	v1.19	Integrar archivos de calidad, mapeo, etc de diferentes análisis	
	Porechop	mamba install -c bioconda porechop	v0.2.4	Eliminar adaptadores	<a href="https://github.com/rrwick/Porechop">https://github.com/rrwick/Porechop</a>
	chopper	mamba install -c bioconda chopper	v0.7.0	Realiza las acciones de NanoFilt + NanoLyse (eliminar regiones de baja calidad y eliminar contaminantes)	<a href="https://github.com/wdecoster/chopper">https://github.com/wdecoster/chopper</a>
	Flye	mamba install -c bioconda flye	v2.9.3	Ensambaje de lecturas largas ONT	<a href="https://github.com/fenderglass/Flye">https://github.com/fenderglass/Flye</a>
	Bandage	mamba install -c bioconda bandage_ng	v2022.09	Visualización del grafo de ensamblaje	Si hay error de instalación en conda, bajar desde web: <a href="https://rrwick.github.io/Bandage/">https://rrwick.github.io/Bandage/</a>
	Minimap2	mamba install -c bioconda minimap2		necesario para polishing	<a href="https://github.com/lh3/minimap2">https://github.com/lh3/minimap2</a>
	Racon	mamba install -c bioconda racon	v1.5.0	polishing	<a href="https://github.com/lcb-sci/racon">https://github.com/lcb-sci/racon</a>
	Medaka	mamba install -c bioconda medaka	v1.8.0	"polishing" de lecturas largas y ensamblaje ONT [medaka_consensus]	Si no podemos instalarlo en el entorno: pip install medaka (instalación en nativo) A veces es necesario instalar a mano bcftools, bgzip, samtools y tabix en las versiones requeridas
	Quast	mamba install -c bioconda quast	v5.2.0	Evaluación de ensamblajes	Para funcionalidad completa es necesario bajar varios accesorios: quast-download-gridss quast-download-silva quast-download-busco [error de descarga, hay que hacer descarga manual]
	Prokka	mamba install -c conda-forge -c bioconda -c defaults prokka	v1.14.6	Anotación de genomas	Antes de utilizarlo <i>puede</i> que sea necesario configurar las variables export PERL5LIB=\$CONDA_PREFIX/lib/perl5/site_perl/5.22.0/ conda env config vars set PERL5LIB=\$CONDA_PREFIX/lib/perl5/site_perl/5.22.0/-n 04MBIF_bacteriano conda deactivate conda activate 04MBIF_bacteriano
	AMRFinderPlus	mamba install -c bioconda ncbi-amrfinderplus	v3.11.26	Detección genes de resistencia a antibióticos, metales pesados y genes de virulencia	Antes de utilizarlo realizar: amrfinder_update
conda env export --file=04MBIF_bacteriano_ONT.yml					





## Tema 5 - Ejemplo 8 - Análisis ONT

En este ejemplo vamos a utilizar unas lecturas obtenidas desde un secuenciador Oxford Nanopore de tipo MinION. Revisaremos su calidad y realizaremos de manera sencilla su ensamblaje y polishing.

1) descargar las lecturas ONT desde [Tema5\\_Ejemplo8\\_Nanopore](#)

2) El primer paso es analizar la calidad de las lecturas secuenciadas. Por ello vamos a utilizar la herramienta NanoPlot, que vamos a instalar en nuestro environment conda 04MBIF\_bacteriano\_ONT. ¿Qué diferencias observas respecto a las lecturas de Illumina?

```
mamba activate 04MBIF_bacteriano_ONT  
  
mamba install -c bioconda nanoplot  
  
NanoPlot -t 2 -o sample1_QC --fastq sample1.nanoporeMdT.fastq.gz
```

3) Realizamos el un filtrado de calidad con NanoFilt (tenemos que instalarlo previamente):

```
mamba install -c bioconda nanofilt  
  
gunzip -c sample1.nanoporeMdT.fastq.gz | NanoFilt --logfile nanofilt.log -l 1000 -q 10 | gzip > trimmed_reads.fastq.gz
```

4) Recuenta cuántas lecturas tenías antes y cuántas ahora. Hazlo en terminal. Recordatorio: grep -c.... (piensa qué sigue)

5) Ensamblamos las lecturas limpias con Flye (ojo, este proceso es bastante lento en nuestra máquina)

```
flye -o genoma --nano-raw trimmed_reads.fastq.gz
```

6) Vamos a realizar una ronda de polishing con Minimap2 y Racon (habitualmente se hacen varias rondas, pero por simplicidad, haremos sólo una)

```
minimap2 -x ava-ont -t 2 genoma/assembly.fasta trimmed_reads.fastq.gz > overlaps1.paf  
  
racon -t 2 trimmed_reads.fastq.gz overlaps1.paf genoma/assembly.fasta > racon1.fasta
```

7) Extraemos la secuencia consenso con Medaka

```
medaka_consensus -i trimmed_reads.fastq.gz -d racon1.fasta -o medaka_sample1 -t 2
```

Nota: si os da error Medaka... "AttributeError: module 'numpy' has no attribute 'typeDict'", instalamos numpy a mano...

```
pip install numpy==1.21
```

6) comprobamos la calidad del ensamblaje con Quast. Indica las diferencias que hay respecto al ensamblaje Illumina.

```
quast.py -o quast_results -m 0 consensus.fasta
```

7) visualizar el grafo con Bandage. ¿Puedes diferenciar plásmidos?

```

usage: NanoPlot [-h] [-v] [-t THREADS] [--verbose] [--raw] [--huge] [-o OUTDIR] [--no_static] [-p PREFIX] [--tsv_stats] [--info_in_report] [--maxlength N] [--minlength N] [--drop_outliers] [--downsample N] [--loglength]
                [--percentqual] [--alength] [--mingual N] [--runtime_until N] [--readtype {1D,2D,1D2}] [--barcoded] [--no_supplementary] [-c COLOR] [-cm COLORMAP]
                [-f [{png,jpg,jpeg,webp,svg,pdf,eps,json} [{png,jpg,jpeg,webp,svg,pdf,eps,json} ...]]) [--plots [{kde,hex,dot} [{kde,hex,dot} ...]]) [--legacy [{kde,dot,hex} [{kde,dot,hex} ...]]) [--listcolors] [--listcolormaps]
                [--no-N50] [-N50] [--title TITLE] [--font_scale FONT_SCALE] [--dpi DPI] [-hide_stats]
                (--fastq file [file ...] | --fasta file [file ...] | --fastq_rich file [file ...] | --fastq_minimal file [file ...] | --summary file [file ...] | --bam file [file ...] | --ubam file [file ...] | --cram file [file ...] | --
pickle pickle | --feather file [file ...])

CREATES VARIOUS PLOTS FOR LONG READ SEQUENCING DATA.

General options:
  -h, --help            show the help and exit
  -v, --version         Print version and exit.
  -t, --threads THREADS
                        Set the allowed number of threads to be used by the script
  --verbose             Write log messages also to terminal.
  --store               Store the extracted data in a pickle file for future plotting.
  --raw                Store the extracted data in tab separated file.
  --huge               Input data is one very large file.
  -o, --outdir OUTDIR  Specify directory in which output has to be created.
  --no_static           Do not make static (png) plots.
  -p, --prefix PREFIX   Specify an optional prefix to be used for the output files.
  --tsv_stats           Output the stats file as a properly formatted TSV.
  --info_in_report      Add NanoPlot run info in the report.

Options for filtering or transforming input prior to plotting:
  --maxlength N        Hide reads longer than length specified.
  --minlength N        Hide reads shorter than length specified.
  --drop_outliers      Drop outlier reads with extreme long length.
  --downsample N       Reduce dataset to N reads by random sampling.
  --loglength          Additionally show logarithmic scaling of lengths in plots.
  --percentqual        Use qualities as theoretical percent identities.
  --alength             Use aligned read lengths rather than sequenced length (bam mode)
  --mingual N          Drop reads with an average quality lower than specified.
  --runtime_until N    Only take the N first hours of a run
  --readtype {1D,2D,1D2}
                        Which read type to extract information about from summary. Options are 1D, 2D,
                        1D2
  --barcoded            Use if you want to split the summary file by barcode
  --no_supplementary   Use if you want to remove supplementary alignments

Options for customizing the plots created:
  -c, --color COLOR    Specify a valid matplotlib color for the plots
  -cm, --colormap COLORMAP
                        Specify a valid matplotlib colormap for the heatmap
  -f, --format [{png,jpg,jpeg,webp,svg,pdf,eps,json} [{png,jpg,jpeg,webp,svg,pdf,eps,json} ...]]
                        Specify the output format of the plots, which are in addition to the html files
  --plots [{kde,hex,dot} [{kde,hex,dot} ...]])
                        Specify which bivariate plots have to be made.
  --legacy [{kde,dot,hex} [{kde,dot,hex} ...]])
                        Specify which bivariate plots have to be made (legacy mode).
  --listcolors          List the colors which are available for plotting and exit.
  --listcolormaps        List the colors which are available for plotting and exit.
  --no-N50              Hide the N50 mark in the read length histogram
  --N50                Show the N50 mark in the read length histogram

```

```

Specify which bivariate plots have to be made.
--legacy [{kde, dot, hex} [{kde, dot, hex} ...]]
                                Specify which bivariate plots have to be made (legacy mode).
--listcolors                  List the colors which are available for plotting and exit.
--listcolormaps                List the colors which are available for plotting and exit.
--no-N50                      Hide the N50 mark in the read length histogram
--N50                          Show the N50 mark in the read length histogram
--title TITLE                 Add a title to all plots, requires quoting if using spaces
--font_scale FONT_SCALE        Scale the font of the plots by a factor
--dpi DPI                     Set the dpi for saving images
--hide_stats                   Not adding Pearson R stats in some bivariate plots

Input data sources, one of these is required.:
--fastq file [file ...]
                                Data is in one or more default fastq file(s).
--fasta file [file ...]
                                Data is in one or more fasta file(s).
--fastq_rich file [file ...]
                                Data is in one or more fastq file(s) generated by albacore, MinKNOW or guppy
                                with additional information concerning channel and time.
--fastq_minimal file [file ...]
                                Data is in one or more fastq file(s) generated by albacore, MinKNOW or guppy
                                with additional information concerning channel and time. Is extracted swiftly
                                without elaborate checks.
--summary file [file ...]
                                Data is in one or more summary file(s) generated by albacore or guppy.
--bam file [file ...]
                                Data is in one or more sorted bam file(s).
--ubam file [file ...]
                                Data is in one or more unmapped bam file(s).
--cram file [file ...]
                                Data is in one or more sorted cram file(s).
--pickle pickle               Data is a pickle file stored earlier.
--feather file [file ...]
                                Data is in one or more feather file(s).

```

**EXAMPLES:**

```

NanoPlot --summary sequencing_summary.txt --loglength -o summary-plots-log-transformed
NanoPlot -t 2 --fastq reads1.fastq.gz reads2.fastq.gz --maxlength 40000 --plots hex dot
NanoPlot --color yellow --bam alignment1.bam alignment2.bam alignment3.bam --downsample 10000

```

# NanoPlot reports

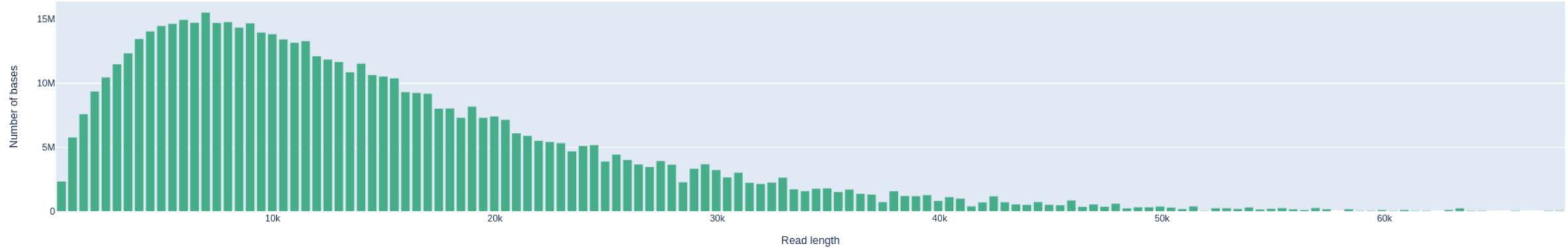
## Summary statistics

General summary	
Mean read length	6,499.7
Mean read quality	15.0
Median read length	4,133.0
Median read quality	15.1
Number of reads	91,086.0
Read length N50	11,845.0
STDEV read length	7,141.2
Total bases	592,035,836.0
Number, percentage and megabases of reads above quality cutoffs	
>Q5	91085 (100.0%) 592.0Mb
>Q7	91079 (100.0%) 592.0Mb
>Q10	90912 (99.8%) 592.0Mb
>Q12	82791 (90.9%) 544.1Mb
>Q15	48034 (52.7%) 314.8Mb
Top 5 highest mean basecall quality scores and their read lengths	
1	30.3 (168)
2	30.2 (21)
3	29.8 (212)
4	28.6 (54)
5	27.7 (356)
Top 5 longest reads and their mean basecall quality score	
1	67874 (12.4)
2	67173 (18.0)
3	65808 (16.6)
4	64080 (14.1)
5	63484 (16.1)

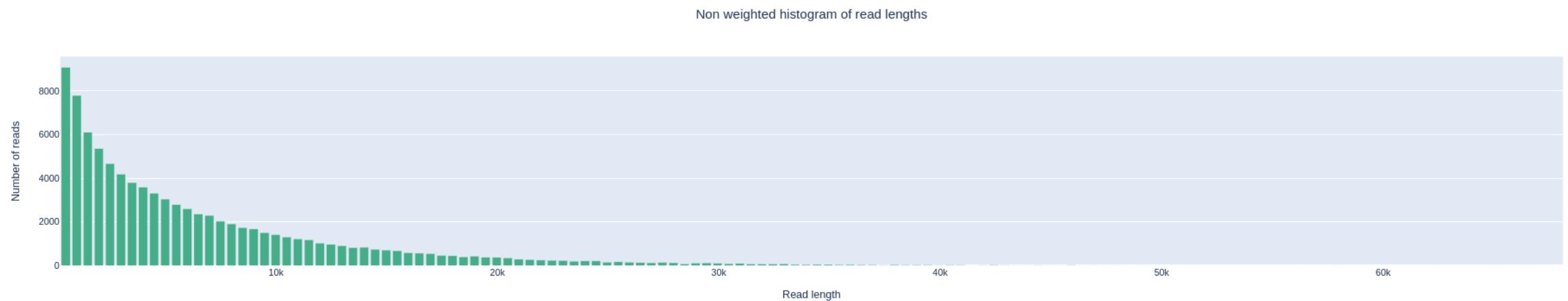
Weighted histogram of read lengths



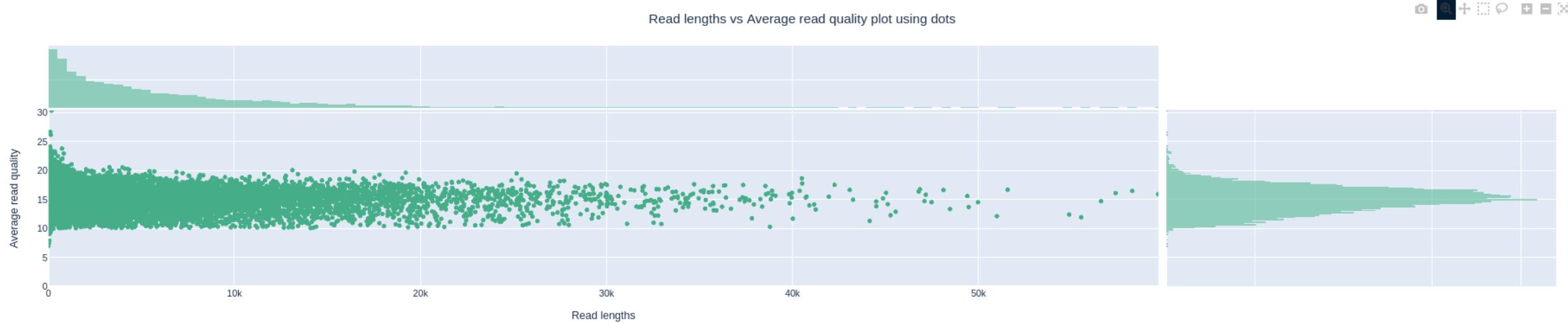
Weighted histogram of read lengths



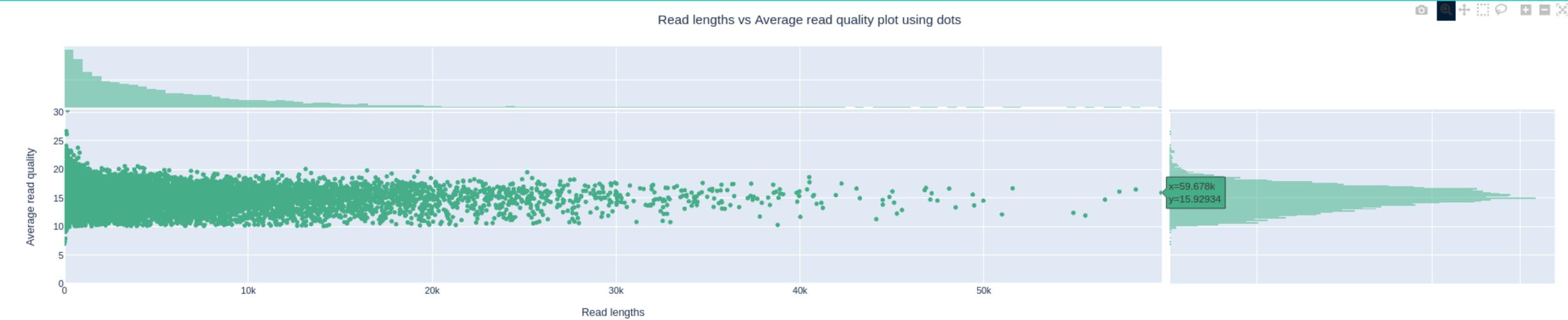
Non weighted histogram of read lengths



Read lengths vs Average read quality plot using dots



Read lengths vs Average read quality plot using dots



## USAGE:

```
NanoFilt [-h] [-v] [--logfile LOGFILE] [-l LENGTH]
          [--maxlength MAXLENGTH] [-q QUALITY] [--minGC MINGC]
          [--maxGC MAXGC] [--headcrop HEADCROP] [--tailcrop TAILCROP]
          [-s SUMMARY] [--readtype {1D,2D,1D2}]
          [input]

Perform quality and/or length and/or GC filtering of (long read) fastq data. Reads on stdin.

General options:
-h, --help      show the help and exit
-v, --version   Print version and exit.
--logfile LOGFILE  Specify the path and filename for the log file.
input           input, uncompressed fastq file (optional)

Options for filtering reads on.:
-l, --length LENGTH  Filter on a minimum read length
--maxlength MAXLENGTH Filter on a maximum read length
-q, --quality QUALITY Filter on a minimum average read quality score
--minGC MINGC    Sequences must have GC content >= to this. Float between 0.0 and 1.0. Ignore
                  using summary file.
--maxGC MAXGC    Sequences must have GC content <= to this. Float between 0.0 and 1.0. Ignore
                  using summary file.

Options for trimming reads.:
--headcrop HEADCROP Trim n nucleotides from start of read
--tailcrop TAILCROP Trim n nucleotides from end of read

Input options.:
-s, --summary SUMMARY Use albacore or guppy summary file for quality scores
--readtype        Which read type to extract information about from summary. Options are 1D, 2D
```

```
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra nanopore_data2]$ zgrep -c 'read=' *.fastq.gz
sample1.nanoporeMdT.fastq.gz:91086
trimmed_reads.fastq.gz:74221
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra nanopore_data2]$ █
```

```
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra nanopore_data2]$ flye -o genoma --nano-raw trimmed_reads.fastq.gz
[2023-01-24 19:32:38] INFO: Starting Flye 2.9.1-b1780
[2023-01-24 19:32:38] INFO: >>>STAGE: configure
[2023-01-24 19:32:38] INFO: Configuring run
[2023-01-24 19:32:57] INFO: Total read length: 583929210
[2023-01-24 19:32:57] INFO: Reads N50/N90: 12017 / 3768
[2023-01-24 19:32:57] INFO: Minimum overlap set to 4000
[2023-01-24 19:32:57] INFO: >>>STAGE: assembly
[2023-01-24 19:32:57] INFO: Assembling disjointigs ....
[2023-01-24 19:32:57] INFO: Reading sequences [2023-01-24 20:13:00] INFO: Median overlap divergence: 0.142857
[2023-01-24 19:33:26] INFO: Counting k-mers: [2023-01-24 20:13:01] INFO: Parsing reads
[2023-01-24 19:33:26] INFO: 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% [2023-01-24 20:13:11] INFO: Aligning reads to the graph
[2023-01-24 19:38:29] INFO: Filling index table [2023-01-24 20:22:34] INFO: Aligned read sequence: 516844680 / 519135342 (0.995588)
[2023-01-24 19:38:29] INFO: 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% [2023-01-24 20:22:34] INFO: Median overlap divergence: 0.0133922
[2023-01-24 19:44:47] INFO: Filling index table [2023-01-24 20:22:34] INFO: Mean edge coverage: 73
[2023-01-24 19:44:47] INFO: 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100% [2023-01-24 20:22:34] INFO: Simplifying the graph
[2023-01-24 19:53:53] INFO: Extending reads [2023-01-24 20:22:35] INFO: >>>STAGE: contigger
[2023-01-24 19:53:53] INFO: [2023-01-24 20:22:35] INFO: Generating contigs
[2023-01-24 19:53:53] INFO: [2023-01-24 20:22:35] INFO: Reading sequences
[2023-01-24 19:53:53] INFO: [2023-01-24 20:22:45] INFO: Generated 3 contigs
[2023-01-24 19:53:53] INFO: [2023-01-24 20:22:45] INFO: Added 0 scaffold connections
[2023-01-24 19:53:53] INFO: [2023-01-24 20:22:45] INFO: >>>STAGE: polishing
[2023-01-24 19:53:53] INFO: [2023-01-24 20:22:45] INFO: Polishing genome (1/1)
[2023-01-24 19:53:53] INFO: [2023-01-24 20:22:45] INFO: Running minimap2
[2023-01-24 19:53:53] INFO: [2023-01-24 20:28:28] INFO: Separating alignment into bubbles
[2023-01-24 19:53:53] INFO: [2023-01-24 20:40:25] INFO: Alignment error rate: 0.033541
[2023-01-24 19:53:53] INFO: [2023-01-24 20:40:25] INFO: Correcting bubbles
[2023-01-24 19:53:53] INFO: [2023-01-24 20:46:35] INFO: >>>STAGE: finalize
[2023-01-24 19:53:53] INFO: [2023-01-24 20:46:36] INFO: Assembly statistics:

Total length: 7058831
Fragments: 3
Fragments N50: 7037522
Largest frg: 7037522
Scaffolds: 0
Mean coverage: 83
```

Tarda > 1 hora

```
(04MBIF_bacteriano_ONT) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra Tema5_ONT]$ minimap2
Usage: minimap2 [options] <target.fa>|<target.idx> [query.fa] [...]
Options:
Indexing:
-H      use homopolymer-compressed k-mer (preferable for PacBio)
-k INT   k-mer size (no larger than 28) [15]
-w INT   minimizer window size [10]
-I NUM    split index for every ~NUM input bases [8G]
-d FILE   dump index to FILE []
Mapping:
-f FLOAT  filter out top FLOAT fraction of repetitive minimizers [0.0002]
-g NUM    stop chain elongation if there are no minimizers in INT-bp [5000]
-G NUM    max intron length (effective with -xsplice; changing -r) [200k]
-F NUM    max fragment length (effective with -xsr or in the fragment mode) [800]
-r NUM[,NUM] chaining/alignment bandwidth and long-join bandwidth [500,20000]
-n INT    minimal number of minimizers on a chain [3]
-m INT    minimal chaining score (matching bases minus log gap penalty) [40]
-X        skip self and dual mappings (for the all-vs-all mode)
-p FLOAT  min secondary-to-primary score ratio [0.8]
-N INT    retain at most INT secondary alignments [5]
Alignment:
-A INT    matching score [2]
-B INT    mismatch penalty (larger value for lower divergence) [4]
-O INT[,INT] gap open penalty [4,24]
-E INT[,INT] gap extension penalty; a k-long gap costs min{O1+k*E1,O2+k*E2} [2,1]
-z INT[,INT] Z-drop score and inversion Z-drop score [400,200]
-s INT    minimal peak DP alignment score [80]
-u CHAR   how to find GT-AG. f:transcript strand, b:both strands, n:don't match GT-AG [n]
-J INT    splice mode. 0: original minimap2 model; 1: miniprot model [1]
Input/Output:
-a        output in the SAM format (PAF by default)
-o FILE   output alignments to FILE [stdout]
-L        write CIGAR with >65535 ops at the CG tag
-R STR    SAM read group line in a format like '@RG\tID:foo\tSM:bar' []
-c        output CIGAR in PAF
--cs[=STR] output the cs tag; STR is 'short' (if absent) or 'long' [none]
--MD      output the MD tag
--eqx     write =/X CIGAR operators
-Y        use soft clipping for supplementary alignments
-t INT    number of threads [3]
-K NUM    minibatch size for mapping [500M]
--version show version number
Preset:
-x STR    preset (always applied before other options)
- map-pb/map-ont - PacBio CLR/Nanopore vs reference
- map-hifi - PacBio HiFi reads vs reference
- ava-pb/ava-ont - PacBio/Nanopore read vs reference
- asm5/asm10/asm20 - asm-to-ref mapping
- splice/splice:hq - long-read/Pacbio-CCS
- sr - genomic short-read mapping
See `man ./minimap2.1` for detailed description of these options.
```

```
(04MBIF_bacteriano_ONT) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra Tema5_ONT]$ minimap2 -x ava-ont -t 2 genoma/assembly.fasta trimmed_reads.fastq.gz > overlaps1.paf
[M::mm_idx_gen::0.444*0.88] collected minimizers
[M::mm_idx_gen::0.709*1.11] sorted minimizers
[M::main::0.710*1.11] loaded/built the index for 2 target sequence(s)
[M::mm_mapopt_update::0.774*1.09] mid_occ = 11
[M::mm_idx_stat] kmer size: 15; skip: 5; is_hpc: 0; #seq: 2
[M::mm_idx_stat::0.823*1.08] distinct minimizers: 2126341 (90.97% are singletons); average occurrences: 1.126; average spacing: 2.943; total length: 7047457
[M::worker_pipeline::104.443*1.79] mapped 63578 sequences
[M::worker_pipeline::120.026*1.80] mapped 10643 sequences
[M::main] Version: 2.26-r1175
[M::main] CMD: minimap2 -x ava-ont -t 2 genoma/assembly.fasta trimmed_reads.fastq.gz
[M::main] Real time: 120.047 sec; CPU: 216.457 sec; Peak RSS: 0.644 GB
(04MBIF_bacteriano_ONT) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra Tema5_ONT]$
```

```
(04MBIF_bacteriano_ONT) [UNIVERSIDADVIU\maria.detoro@a-a8i1nqzd50ra Tema5_ONT]$ racon -h
usage: racon [options ...] <sequences> <overlaps> <target sequences>
```

```
#default output is stdout
<sequences>
  input file in FASTA/FASTQ format (can be compressed with gzip)
  containing sequences used for correction
<overlaps>
  input file in MHAP/PAF/SAM format (can be compressed with gzip)
  containing overlaps between sequences and target sequences
<target sequences>
  input file in FASTA/FASTQ format (can be compressed with gzip)
  containing sequences which will be corrected

options:
  -u, --include-unpolished
    output unpolished target sequences
  -f, --fragment-correction
    perform fragment correction instead of contig polishing
    (overlaps file should contain dual/self overlaps!)
  -w, --window-length <int>
    default: 500
    size of window on which POA is performed
  -q, --quality-threshold <float>
    default: 10.0
    threshold for average base quality of windows used in POA
  -e, --error-threshold <float>
    default: 0.3
    maximum allowed error rate used for filtering overlaps
  --no-trimming
    disables consensus trimming at window ends
  -m, --match <int>
    default: 3
    score for matching bases
  -x, --mismatch <int>
    default: -5
    score for mismatching bases
  -g, --gap <int>
    default: -4
    gap penalty (must be negative)
  -t, --threads <int>
    default: 1
    number of threads
  --version
    prints the version number
  -h, --help
    prints the usage
```

```
(04MBIF_bacteriano_ONT) [UNIVERSIDADVIU\maria.detoro@a-a8i1nqzd50ra Tema5_ONT]$ racon -t 2 trimmed_reads.fastq.gz overlaps1.paf genoma/assembly.fasta > racon1.fasta
[racon::Polisher::initialize] loaded target sequences 0.070648 s
[racon::Polisher::initialize] loaded sequences 16.919371 s
[racon::Polisher::initialize] loaded overlaps 0.195525 s
[racon::Polisher::initialize] aligning overlaps [=====] 161.812127 s
[racon::Polisher::initialize] transformed data into windows 1.443464 s
[racon::Polisher::polish] generating consensus [=====] 371.195667 s
[racon::Polisher::] total = 551.750022 s
```

racon1.fasta

```
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra nanopore_data]$ medaka_consensus
medaka 1.7.2
-----
Assembly polishing via neural networks. Medaka is optimized
to work with the Flye assembler.

medaka_consensus [-h] -i <fastx> -d <fasta>

-h show this help text.
-i fastx input basecalls (required).
-d fasta input assembly (required).
-o output folder (default: medaka).
-g don't fill gaps in consensus with draft sequence.
-r use gap-filling character instead of draft sequence (default: None)
-m medaka model, (default: r941_min_hac_g507).
Choices: r103_fast_g507 r103_hac_g507 r103_min_high_g345 r103_min_high_g360 r103_prom_high_g360 r103_sup_g507 r1041_e82_260bps_fast_g632 r1041_e82_260bps_hac_g632 r1041_e82_260bps_sup_g632 r1041_e82_400bps_fast_g615 r1041_e82_400bps_fast_g632 r1041_e82_400bps_hac_g615 r1041_e82_400bps_hac_g632 r1041_e82_400bps_sup_g615 r104_e81_fast_g5015 r104_e81_hac_g5015 r104_e81_sup_g5015 r104_e81_sup_g610 r10_min_high_g303 r10_min_high_g340 r941_e81_fast_g514 r941_e81_hac_g514 r941_e81_sup_g514 r941_min_fast_g303 r941_min_fast_g507 r941_min_hac_g507 r941_min_high_g303 r941_min_high_g330 r941_min_high_g340_rle r941_min_high_g344 r941_min_high_g351 r941_min_high_g360 r941_min_sup_g507 r941_prom_fast_g303 r941_prom_fast_g507 r941_prom_hac_g507 r941_prom_high_g303 r941_prom_high_g330 r941_prom_high_g344 r941_prom_high_g360 r941_prom_high_g4011 r941_prom_sup_g507 r941_sup_plant_g610
    Alternatively a .tar.gz/.hdf file from 'medaka train'.
-f Force overwrite of outputs (default will reuse existing outputs).
-x Force recreation of alignment index.
-t number of threads with which to create features (default: 1).
-b batchsize, controls memory use (default: 100).

-i must be specified.
```

## Models

For best results it is important to specify the correct model, `-m` in the above, according to the basecaller used. Allowed values can be found by running `medaka tools list\models`.

Medaka models are named to indicate i) the pore type, ii) the sequencing device (MinION or PromethION), iii) the basecaller variant, and iv) the basecaller version, with the format:

```
{pore}_{device}_{caller variant}_{caller version}
```

For example the model named `r941_min_fast_g303` should be used with data from MinION (or GridION) R9.4.1 flowcells using the fast Guppy basecaller version 3.0.3. By contrast the model `r941_prom_hac_g303` should be used with PromethION data and the high accuracy basecaller (termed "hac" in Guppy configuration files). Where a version of Guppy has been used without an exactly corresponding medaka model, the medaka model with the highest version equal to or less than the guppy version should be selected.

```
[22:16:01 - Sampler] Took 0.06s to make features.
[22:16:05 - Feature] Processed contig_1:5994000.0-6993999.0 (median depth 101.0)
[22:16:05 - Sampler] Took 8.00s to make features.
[22:17:28 - PWorker] Batches in cache: 8.
[22:17:28 - PWorker] 8.7% Done (0.6/7.1 Mbases) in 119.8s
[22:18:46 - PWorker] Batches in cache: 8.
[22:18:46 - PWorker] 17.2% Done (1.2/7.1 Mbases) in 198.0s
[22:20:24 - PWorker] Batches in cache: 8.
[22:20:24 - PWorker] 25.6% Done (1.8/7.1 Mbases) in 295.6s
[22:22:46 - PWorker] Batches in cache: 8.
[22:22:46 - PWorker] 34.4% Done (2.4/7.1 Mbases) in 438.0s
[22:23:28 - PWorker] Batches in cache: 8.
[22:23:28 - PWorker] 43.4% Done (3.1/7.1 Mbases) in 479.8s
[22:24:09 - PWorker] Batches in cache: 7.
[22:24:09 - PWorker] 52.3% Done (3.7/7.1 Mbases) in 520.9s
[22:24:36 - PWorker] Batches in cache: 6.
[22:24:36 - PWorker] 60.9% Done (4.3/7.1 Mbases) in 547.7s
[22:25:03 - PWorker] Batches in cache: 5.
[22:25:03 - PWorker] 69.1% Done (4.9/7.1 Mbases) in 574.9s
[22:25:31 - PWorker] Batches in cache: 4.
[22:25:31 - PWorker] 77.3% Done (5.5/7.1 Mbases) in 602.4s
[22:25:57 - PWorker] Batches in cache: 3.
[22:25:57 - PWorker] 85.3% Done (6.0/7.1 Mbases) in 628.9s
[22:26:38 - PWorker] Batches in cache: 2.
[22:26:38 - PWorker] 93.5% Done (6.6/7.1 Mbases) in 670.0s
[22:27:21 - PWorker] Batches in cache: 1.
[22:27:21 - PWorker] 100.0% Done (7.1/7.1 Mbases) in 712.8s
[22:27:22 - PWorker] Processed 12 batches
[22:27:22 - PWorker] All done, 0 remainder regions.
[22:27:22 - Predict] Finished processing all regions.
[22:27:29 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:29 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:29 - DataIndx] Loaded 1/1 (100.00%) sample files.
/home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/lib/python3.8/site-packages/medaka/labels.py:387: RuntimeWarning: divide by zero encountered in log10
    q = -10 * np.log10(err)
[22:27:30 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:31 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:32 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:33 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:34 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:35 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:36 - DataIndx] Loaded 1/1 (100.00%) sample files.
[22:27:36 - DataIndx] Loaded 1/1 (100.00%) sample files.
Polished assembly written to medaka_out/consensus.fasta, have a nice day.
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra nanopore_data2]$ █
```

```
(04MBIF_bacteriano) [UNIVERSIDADVIU\maria.detoro@a-a8ilnqzd50ra nanopore_data2]$ medaka_consensus -i trimmed_reads.fastq.gz -d genoma/assembly.fasta -o medaka_out
Checking program versions
This is medaka 1.7.2
Program Version Required Pass
bcftools 1.16 1.11 True
bgzip 1.16 1.11 True
minimap2 2.24 2.11 True
samtools 1.16.1 1.11 True
tabix 1.16 1.11 True
Aligning basecalls to draft
Creating fai index file /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/genoma/assembly.fasta.fai
Creating mmi index file /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/genoma/assembly.fasta.map-ont.mmi
[M::mm_idx_gen::0.636*0.60] collected minimizers
[M::mm_idx_gen::0.909*0.71] sorted minimizers
[M::main::1.112*0.68] loaded/built the index for 3 target sequence(s)
[M::mm_idx_stat] kmer size: 15; skip: 10; is_hpc: 0; #seq: 3
[M::mm_idx_stat::1.145*0.68] distinct minimizers: 1174782 (91.27% are singletons); average occurrences: 1.121; average spacing: 5.361; total length: 7058831
[M::main] Version: 2.24-r1122
[M::main] CMD: minimap2 -I 16G -x map-ont -d /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/genoma/assembly.fasta.map-ont.mmi /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/genoma/assembly.fasta
[M::main] Real time: 1.161 sec; CPU: 0.789 sec; Peak RSS: 0.070 GB
[M::main::0.158*0.66] loaded/built the index for 3 target sequence(s)
[M::mm_mapopt_update::0.215*0.62] mid_occ = 11
[M::mm_idx_stat] kmer size: 15; skip: 10; is_hpc: 0; #seq: 3
[M::mm_idx_stat::0.266*0.58] distinct minimizers: 1174782 (91.27% are singletons); average occurrences: 1.121; average spacing: 5.361; total length: 7058831
[M::worker_pipeline::692.174*0.50] mapped 63578 sequences
[M::worker_pipeline::796.885*0.51] mapped 10643 sequences
[M::main] Version: 2.24-r1122
[M::main] CMD: minimap2 -x map-ont --secondary=no -L --MD -A 2 -B 4 -O 4,24 -E 2,1 -t 1 -a /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/genoma/assembly.fasta.map-ont.mmi /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/trimmed_reads.fastq.gz
[M::main] Real time: 796.914 sec; CPU: 403.538 sec; Peak RSS: 1.189 GB
[bam_sort_core] merging from 1 files and 1 in-memory blocks...
Running medaka consensus
[22:15:25 - Predict] Setting tensorflow inter/intra-op threads to 1/1.
[22:15:25 - Predict] Processing region(s): contig_1:0-7037522 contig_2:0-9911 contig_3:0-11398
[22:15:25 - Predict] Using model: /home/maria.detoro/miniconda3/envs/04MBIF_bacteriano/lib/python3.8/site-packages/medaka/data/r941_min_hac_g507_model.tar.gz.
[22:15:25 - Predict] Processing 10 long region(s) with batching.
[22:15:28 - MdlStrTF] Model <keras.engine.sequential.Sequential object at 0x7f6c8330af0>
[22:15:28 - MdlStrTF] loading weights from /tmp/tmpbexi_ooh/model/variables/variables
[22:15:28 - BAMFile] Creating pool of 16 BAM file sets.
[22:15:28 - Sampler] Initializing sampler for consensus of region contig_1:0-1000000.
[22:15:28 - Sampler] Initializing sampler for consensus of region contig_1:999000-1999000.
[22:15:28 - PWorker] Running inference for 7.1M draft bases.
[22:15:38 - Feature] Processed contig_1:999000.0-1998999.0 (median depth 74.0)
[22:15:38 - Sampler] Took 9.59s to make features.
[22:15:38 - Sampler] Initializing sampler for consensus of region contig_1:1998000-2998000.
[22:15:38 - Feature] Processed contig_1:0.0-999999.0 (median depth 88.0)
[22:15:38 - Sampler] Took 10.11s to make features.
[22:15:38 - Sampler] Initializing sampler for consensus of region contig_1:2997000-3997000.
```

```
03. Linux-5.15.7-51.158.ami21bz.x86_64-x86_64-with-glibc2.10 (Linux_64)
Python version: 3.8.15
CPUs number: 2

Started: 2023-01-24 22:39:46

Logging to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/quast.log
NOTICE: Maximum number of threads is set to 1 (use --threads option to set it manually)

CWD: /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out
Main parameters:
  MODE: default, threads: 1, min contig length: 0, min alignment length: 65, min alignment IDY: 95.0, \
  ambiguity: one, min local misassembly length: 200, min extensive misassembly length: 1000

Contigs:
  Pre-processing...
  consensus.fasta ==> consensus

2023-01-24 22:39:47
Running Basic statistics processor...
  Contig files:
    consensus
  Calculating N50 and L50...
    consensus, N50 = 7038044, L50 = 1, auN = 7016799.9, Total length = 7059385, GC % = 65.90, # N's per 100 kbp = 0.00
  Drawing Nx plot...
    saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/basic_stats/Nx_plot.pdf
  Drawing cumulative plot...
    saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/basic_stats/cumulative_plot.pdf
  Drawing GC content plot...
    saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/basic_stats/GC_content_plot.pdf
  Drawing consensus GC content plot...
    saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/basic_stats/consensus_GC_content_plot.pdf
Done.

NOTICE: Genes are not predicted by default. Use --gene-finding or --glimmer option to enable it.

2023-01-24 22:39:49
Creating large visual summaries...
This may take a while: press Ctrl-C to skip this step..
  1 of 2: Creating PDF with all tables and plots...
  2 of 2: Creating Icarus viewers...
Done

2023-01-24 22:39:50
RESULTS:
  Text versions of total report are saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/report.txt, report.tsv, and report.tex
  Text versions of transposed total report are saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/transposed_report.txt, transposed_report.tsv, and transposed_report.tex
  HTML version (interactive tables and plots) is saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/report.html
  PDF version (tables and plots) is saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/report.pdf
  Icarus (contig browser) is saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/icarus.html
  Log is saved to /home/maria.detoro/04MBIF/Tema5_bacterianos/nanopore_data2/medaka_out/quast_results/quast.log

Finished: 2023-01-24 22:39:50
Elapsed time: 0:00:03.424160
NOTICES: 2; WARNINGS: 0; non-fatal ERRORs: 0

Thank you for using QUAST!
```

[View in Icarus contig browser](#)

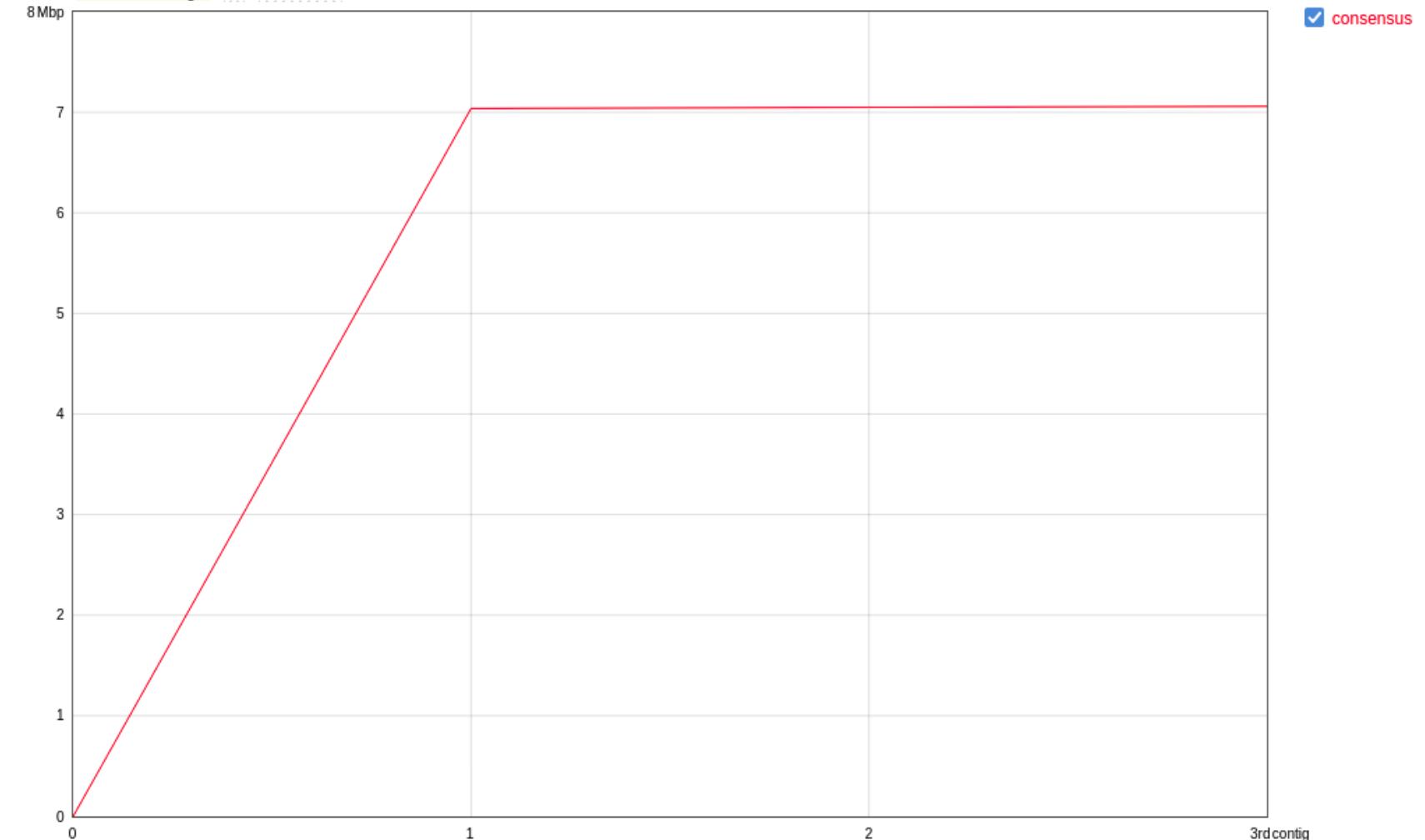
All statistics are based on contigs of size  $\geq 0$  bp, unless otherwise noted (e.g., "# contigs ( $\geq 0$  bp)" and "Total length ( $\geq 0$  bp)" include all contigs).

**Statistics without reference**  consensus

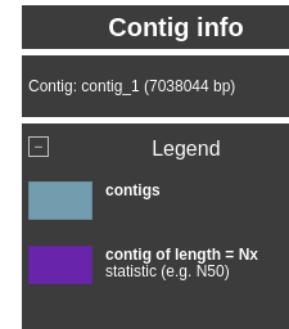
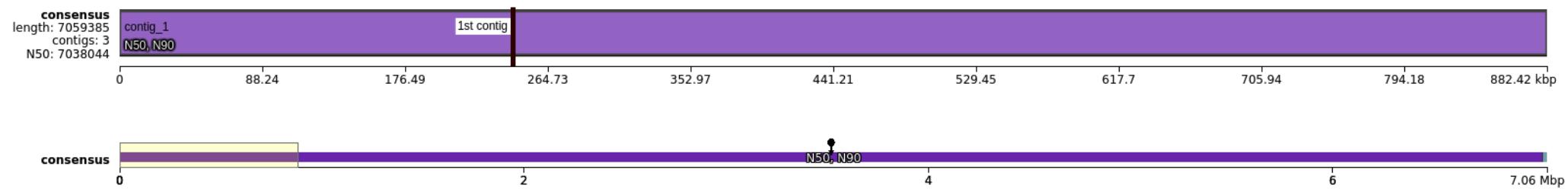
# contigs	3
# contigs ( $\geq 0$ bp)	3
# contigs ( $\geq 1000$ bp)	3
# contigs ( $\geq 5000$ bp)	3
# contigs ( $\geq 10000$ bp)	2
# contigs ( $\geq 25000$ bp)	1
# contigs ( $\geq 50000$ bp)	1
Largest contig	7 038 044
Total length	7 059 385
Total length ( $\geq 0$ bp)	7 059 385
Total length ( $\geq 1000$ bp)	7 059 385
Total length ( $\geq 5000$ bp)	7 059 385
Total length ( $\geq 10000$ bp)	7 049 473
Total length ( $\geq 25000$ bp)	7 038 044
Total length ( $\geq 50000$ bp)	7 038 044
N50	7 038 044
N90	7 038 044
auN	7 016 800
L50	1
L90	1
GC (%)	65.9

**Mismatches**

# N's per 100 kbp	0
# N's	0

**Plots:** Cumulative length Nx GC content

Contigs are ordered from largest (contig #1) to smallest.

**Contig size viewer**


File Edit Tools View Select Output Help

### Graph information

Nodes: 0  
Edges: 0  
Paths: 0  
Total length: 0

[More info](#)

### Graph drawing

Scope: Entire graph  
Style: Single  Double   
[Draw graph](#)

### Graph display

Zoom: 100.0%  
Node width: 5.0  
[Random colours](#)

### Node labels

Custom  Name  
 Length  Depth  
 CSV data:  
[Font](#)  Text outline

- › Graph search
- › BED
- › Annotations

Load graph

Look in: /home/maria.detoro/04...nanopore\_data2/genoma

Name	Type	Date Modified
00-assembly	Folder	1/24/23 8:01 PM
10-consensus	Folder	1/24/23 8:12 PM
20-repeat	Folder	1/24/23 8:22 PM
30-contigger	Folder	1/24/23 8:22 PM
40-polishing	Folder	1/24/23 8:46 PM
assembly_graph.gfa	6.73 MiB plain text	1/24/23 8:46 PM
assembly_graph.gv	689 bytes GraphViz	1/24/23 8:46 PM
assembly_info.txt	155 bytes plain text	1/24/23 8:46 PM
assembly.fasta	6.84 MiB plain text	1/24/23 8:46 PM
assembly.fasta.fai	83 bytes plain text	1/24/23 10:00 PM
assembly.fasta.map-ont.mmi	23.28 MiB unknown	1/24/23 10:00 PM
flye.log	40.49 KiB application log	1/24/23 8:46 PM
params.json	92 bytes JSON	1/24/23 8:46 PM

File name: assembly\_graph.gfa [Open](#)

Files of type: Any supported graph (\*) [Cancel](#)

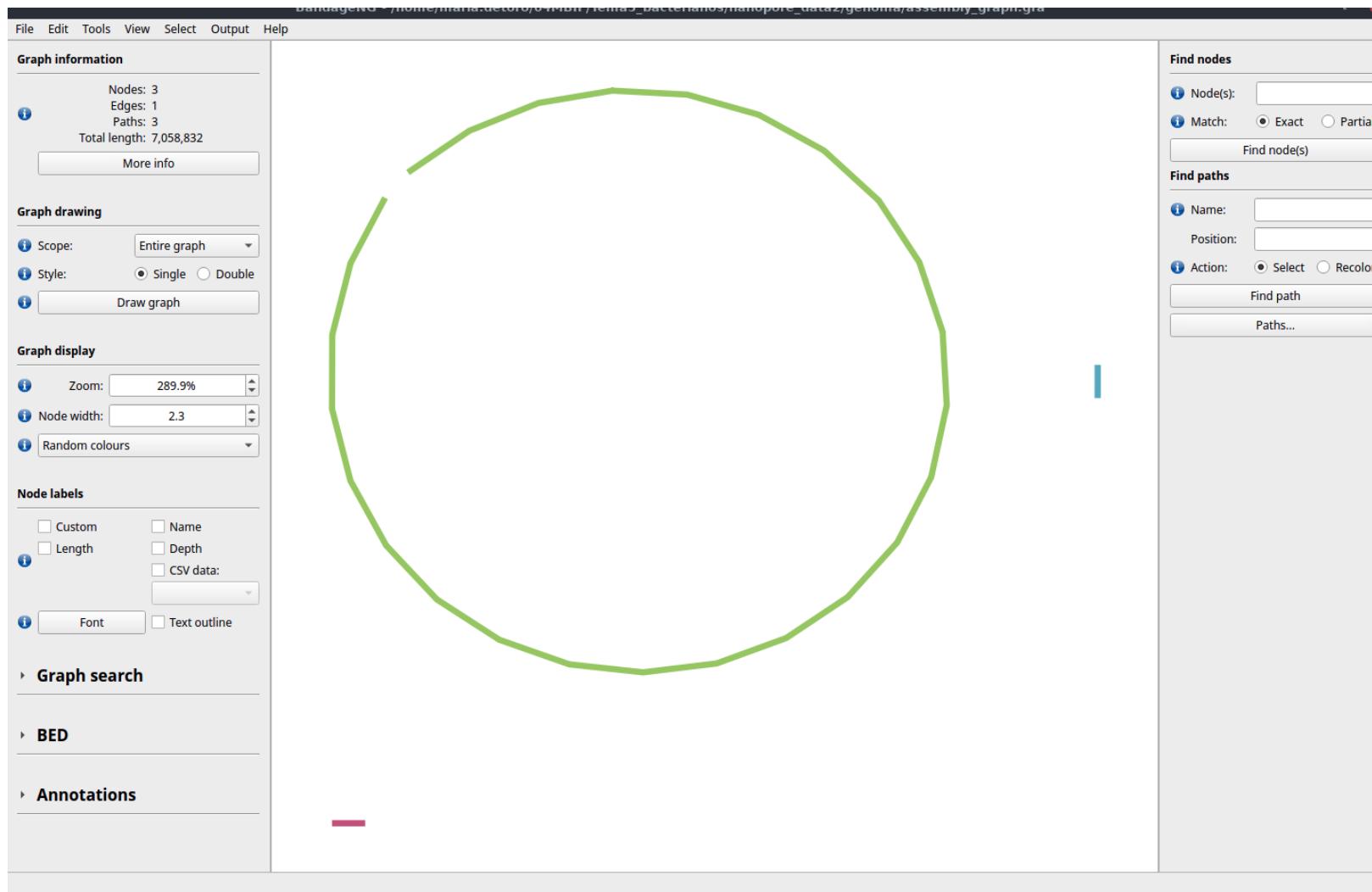
### Find nodes

Node(s):   
 Match:  Exact  Partial  
[Find node\(s\)](#)

### Find paths

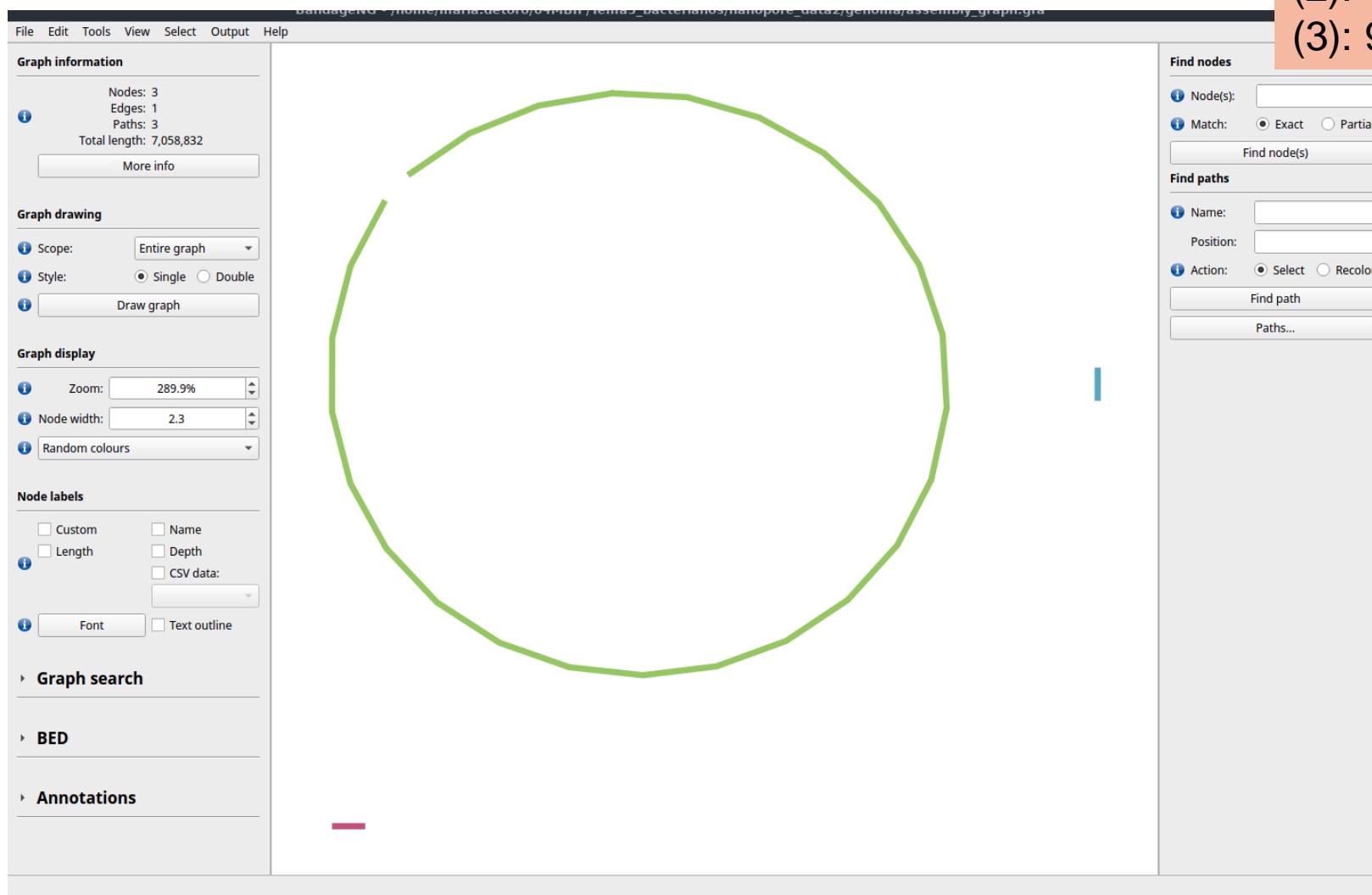
Name:   
Position:   
 Action:  Select  Recolor  
[Find path](#)  
[Paths...](#)

¿Cuántos fragmentos hay? ¿Cuál es la longitud total ensamblada?



¿Cuántos fragmentos hay? ¿Cuál es la longitud total ensamblada?

- (1): 7,037,522 bp; 65,9x  
 (2): 11,398 bp; 64,9%  
 (3): 9,912 bp; 64,9%



- (1): 7,037,522 bp; 65,9x
- (2): 11,398 bp; 64,9%
- (3): 9,912 bp; 64,9%

```
awk -F "|" '/^>/ {close(F); ID=$1; gsub("^>", "", ID); F=ID".fasta"} {print >> F}' consensus.fasta
```

# ¡Gracias!

The logo consists of the lowercase letters "viu" in white, centered within a dark orange, rounded rectangular shape.

viu

**Universidad**  
Internacional  
de Valencia

[universidadviu.com](http://universidadviu.com)

De:  
 Planeta Formación y Universidades