Universitat Oberta de Catalunya (UOC)

Máster Universitario en Bioinformática y Bioestadística

TRABAJO FINAL DE MÁSTER

Subárea 12: Biología molecular y genética

# Desarrollo de herramientas bioinformáticas en Python y R para el análisis de duplicaciones génicas en bacterias y visualización de datos.

---

Autor: Alba Moya Garcés

Tutor: José F. Sánchez Herrero

Profesor: Ferran Prados

---

Barcelona, 5 de enero de 2021

# Créditos/Copyright

# Ficha del Trabajo Final

| | |
|---|---|
| Título del trabajo: | Desarrollo de herramientas bioinformáticas en Python y R para el análisis de duplicaciones génicas en bacterias y visualización de datos |
| Nombre del autor: | Alba Moya Garcés |
| Nombre del colaborador/a docente: | José F. Sánchez Herrero |
| Nombre del PRA: | Ferran Prados |
| Fecha de entrega (mm/aaaa): | 5 de enero de 2020 |
| Titulación o programa: | Máster de Bioinformática y Bioestadística |
| Área del Trabajo Final: | Biología molecular y genética |
| Idioma del trabajo: | Español |
| Palabras clave | microbiología, duplicación génica, grupo ESKAPE |

# Dedicatoria/Cita

Le dedico este trabajo a mi madre, que siempre me apoyó en todo y siempre está ahí sin estarlo.

# Agradecimientos

Quiero agradecer especialmente la oportunidad brindada a Jose para realizar este proyecto, me has metido en una aventura de códigos, versiones y lenguajes raros de la que será difícil salir. Gracias por tanta paciencia y por demostrar empatía cada vez que surgía algún imprevisto derivado de cualquier cosa loca que le pasa a las madres que quieren ampliar sus estudios.

A Roi y a Sergi, los amores de mi vida. Vosotros habéis conseguido poner a prueba mi paciencia y los niveles de estrés que puedo llegar a tolerar. Gracias por permitirme demostrar que uno se puede concentrar aunque el mundo se esté derrumbando del otro lado de la puerta.

Rober, mi compañero de camino, has aceptado más o menos silenciosamente periodos de paternidad en solitario cuando tenía que hacer alguna entrega. Gracias por entretener a las fieras, alimentarlas y mantenerlas con vida mientras yo le daba a la tecla

Ay, Berta, ¿qué conjuro hemos invocado para que de una forma u otra siempre crucemos nuestros caminos? Se nos ocurrió a la vez la genialidad de hacer un máster en bioinformática y aquí estamos, después de mucho sufrimiento, ¡lo hemos conseguido! Muchas gracias por tantas tardes de tecleo compartido, por tu ayuda aún en la distancia, volveremos a encontrarnos.

Y a todos a los que he fallado durante estos años de máster a tiempo parcial. Alba vuelve a las calles con más energías que nunca.

No puedo olvidarme de ese virusito que anda haciendo de las suyas. Quedará raro agradecerle nada, pero, aparte de provocar que me cierren las salas de estudio de la biblioteca, que haya limitaciones de movimiento y socialización ha favorecido tremendamente que acabara el máster. Ahora ya te puedes marchar, que quiero hacer cosas de persona normal.

Y seguro que me olvido de alguien, he tardado tanto en hacer este máster que alguno habrá que me haya ayudado de una forma u otra, gracias.

# Resumen

Las bacterias del grupos ESKAPE son especialmente patógenas para los seres humanos debido a la existencia de cepas multirresistentes a antibióticos. Estudios de duplicidades génicas en los genomas de especies de este grupo y *E. coli* mostraron patrones diferenciados entre cepas virulentas y aquellas que no lo eran, lo cual supone una información muy valiosa para hacer frente a las enfermedades desarrolladas. El objetivo de este TFM trata de completar las investigaciones previas mediante el desarrollo de las herramientas bioinformáticas necesarias para llevar a cabo este proceso de una forma más eficiente e intuitiva. Se han implementado dos programas en python y R que permiten tomar los datos contenidos en un fichero de anotación dado, analizarlos y realizar todos los pasos necesarios hasta la consecución de un archivo con la anotación de las proteínas duplicadas encontradas y una representación gráfica de las mismas. Finalmente, se han seleccionado un pequeño grupo de cepas de las especies del grupo ESKAPE no estudiadas previamente y se ha utilizado como ejemplo de uso de las herramientas. De estos ejemplos de uso se han obtenido datos que podrían ser analizados posteriormente en busca de patrones o particularidades relevantes.

**Palabras clave**: microbiología, duplicación génica, grupo ESKAPE

# Abstract

ESKAPE bacteria are thoght tob be especially resistant to antibiotics and can be particularly dangerous in people with weakened immune systems. Studies of gene duplication in the genomes of ESKAPE specia and *E. coli* shown differentiated patterns between virulent strains and those that were not, which is a very valuable information that can be useful to struggle cronical diseases. The aim of this TFM is to supplement the previous research works and develops bioinformatics tools in order to carry out a process in a more efficient and intuitive way. Two programs have been implemented on python and R that allow process an annotation file, analize it and carry out all the necessary steps until a annotation file of the duplicated proteins found and plot it. Finally, a small group of selected strains for each ESKAPE species not previously studied has been used as an usage example of the generated tools. DAta obtained could be subsequently analyzed in search of relevant patterns or particularities.

**Keywords**: microbiology, gene duplication, ESKAPE pathogens

# Índice general

# Índice de figuras

# Capítulo 1

# Introducción

## 1.1. Contexto y justificación del trabajo

Durante el desarrollo de este trabajo de fin de máster (TFM) se ha llevado a cabo una caracterización de las posibles duplicidades génicas presentes en los genomas de diferentes cepas bacterianas entre las especies del grupo ESKAPE no analizadas en estudios previos *(Klebsiella pneumoniae, Acinetobacter baumannii, Pseudomonas aeruginosa* y *Enterobacter* spp.). Para ello se implementarán diversos módulos y paquetes específicos en los lenguajes de programación Python y R que realizarán de manera automática la búsqueda de las mismas. Con los resultados de la obtenidos, se podrán caracterizar genes específicos relacionados con virulencia o resistencia a antimicrobianos y generar gráficas para su visualización.

La duplicación genética es un proceso que se encuentra tanto en células eucariotas como procariotas y es uno de los tipos de mutación más frecuentes [6, 4]. Es un mecanismo que genera una o más copias indistinguibles del mismo gen en el genoma celular y que se mantiene a lo largo de la línea evolutiva [5, 8, 15, 4]. La ventaja evolutiva que proporciona para las bacterias mantener en sus genomas diferentes copias para el mismo gen viene determinada por que cada una de ellas podría mostrar diferentes patrones de expresión e, incluso, responder ante estímulos diferentes, lo cual dota a las cepas que portan esta variedad de copias de más oportunidades para adaptarse a variaciones fisicoquímicas del medio en el que se encuentran. El aumento de la complejidad genómica que genera esta diversidad funcional, facilita la proliferación de las células ante medios limitantes e incrementa las probabilidades de nuevas mutaciones adaptativas [6, 4].

Las bacterias del grupo ESKAPE son especialmente patógenas para los seres humanos ya que presentan en su material genético genes de resistencia a diferentes agentes antimicrobianos, lo que las hace difíciles de combatir en caso de infección [8, 13]. La existencia de cepas patógenas multirresistentes a diferentes antibióticos ha supuesto un problema sanitario que lleva a graves consecuencias en la salud pública [14, 3, 9, 15]. La limitación de los grupos científicos para crear nuevos antibióticos efectivos y la rapidez con que estas bacterias evolucionan, hace necesario aumentar los conocimientos alrededor de cómo se interrelacionan los diferentes genes bacterianos al expresarse y favorecer la supervivencia de la bacteria [14, 9, 13].

El estudio de duplicidades génicas en *E. coli*, estafilococos y enterococos ha demostrado un patrón diferenciado de las mismas entre las cepas más virulentas y las que no presentan patogenicidad. Se ha constatado la presencia de un número significativo de genes duplicados en cepas patógenas, respecto a aquellas no patógenas, que no se muestran en cepas inofensivas, lo que sugiere que estos genes podrían favorecer de forma relevante la virulencia de las mismas [14, 15].

La duplicación genética en bacterias se produce más frecuentemente por transferencia horizontal genética (HGT) entre diferentes individuos gracias a agentes virales, plásmidos o transposones, que por diferentes mecanismos moleculares naturales que pueden generar duplicaciones en los genomas bacterianos [11, 2, 15, 13].

Identificar genes duplicados específicos en diferentes cepas patógenas, podría contribuir al estudio de los mecanismos de la virulencia de las mismas ampliando la información ya conocida sobre cómo se regula la expresión genética en las bacterias a estudio [14, 15]. Por otro lado, si se determinara la codificación por parte de alguno de estos genes duplicados de productos antigénicos, se podrían desarrollar nuevas vacunas o medicamentos más efectivos para hacer frente a las enfermedades provocadas por este tipo de bacterias [14, 6].

Por todo lo mencionado, en este Trabajo de TFM se complementarán los trabajos realizados por [14] y [15] desarrollando una herramienta bioinformática capaz de llevar a cabo de manera automatizada todo el proceso de búsqueda e identificación de duplicidades dentro de un genoma dado. Se obtendría como resultado un archivo de datos con toda la información obtenida y una representación gráfica con el fin de facilitar su estudio y comprensión. Estos resultado podrán utilizarse fácilmente en estudios de investigación relacionados.

Desarrollar una herramienta bioinformática, compuesta de diversos módulos específicos para cada una de las fases que comprenden el análisis genómico y la búsqueda de duplicidades, permitirá a la comunidad científica avanzar de forma más ágil en la investigación sobre el tema.

## 1.2. Objetivos del Trabajo

Los objetivos desarrollados a lo largo de este TFM han sido los siguientes:

- Desarrollo de herramientas bioinformáticas para el análisis y caracterización de duplicidades genéticas en bacterias.
  - Desarrollar funciones de python para el tratamiento y análisis de duplicaciones tanto de secuencias genómicas en bases de datos como ensambladas *de novo* por parte del usuario.
  - Implementar la generación y representación de resultados para la mejor interpretación y visualización de los datos.
- Caracterización de duplicaciones génicas en una selección de cepas bacterianas.
  - Selección de cepas del grupo ESKAPE para caracterizar duplicaciones génicas.
  - Analizar las secuencias genómicas de las cepas seleccionadas y localizar e identificar genes duplicados

## 1.3. Enfoque y método seguido

A pesar de que ya existen trabajos previos en los cuales se muestran técnicas para el análisis de duplicidades génicas [14, 15], estos no presentan un desarrollo de herramientas bioinformáticas que permitan automatizar totalmente el proceso de análisis y posterior interpretación de los resultados. La estrategia que se llevará a cabo en este TFM procurará una única metodología de análisis genómico que permita a la comunidad científica realizar este proceso de una forma más directa. Cada fase del análisis, desde la lectura de los datos hasta la visualización e interpretación de resultados, se englobará en un único proceso, lo que facilitará el uso y difusión de las herramientas generadas.

En primer lugar, se han desarrollado módulos en Python [28] para leer de manera automática la secuencia genómica proporcionada por el usuario y analizar la información indicada. Posteriormente, con la herramienta de análisis de secuencias BLAST+ [21], se ha implementado otros módulos para buscar e identificar genes duplicados y generar un archivo de resultados. Los diferentes módulos creados en python se han integrado en una única herramienta que trabajará de manera automática a lo largo de todos los pasos requeridos para completar el análisis.

Finalmente, se ha creado un paquete en R [26] para mostrar los resultados de manera interactiva con el programa BioCircos [17] que facilitan su interpretación. Este módulo se ha generado a partir del aportado por [15] con las modificaciones pertinentes para adaptarlo a nuestras necesidades.

Una vez desarrollada esta versión beta del programa, se ha desarrollado el segundo objetivo y analizado cepas bacterianas reales para generar resultados.

## 1.4. Planificación del Trabajo

Los objetivos y tareas realizados se han planteado según los diferentes hitos marcados por el plan docente. Estos hitos corresponden a cada una de las pruebas de evaluación continua (PEC) propuestas a lo largo del periodo estipulado, así como la entrega de la memoria final y la defensa pública del proyecto por medio de una presentación virtual.

La imagen 1.1 representa la planificación del TFM. Se ha desglosado cada objetivo en las correspondientes tareas a realizar a lo largo de la línea temporal.

A continuación, se presenta un breve resumen de algunas de las tareas realizadas más significativas:

- **Creación de un entorno virtual en Python:** para poder realizar correctamente las tareas de programación, se ha creado un espacio único para el proyecto en el que se albergue las diferentes librerías instaladas. El manejo del programa de instalación *pip* ha sido fundamental para el desarrollo de este entorno. Así como el uso de aplicaciones para el control de versiones como GIT y el trabajo colaborativo en plataformas virtuales como GitHub ha favorecido la fluidez a la hora de realizar correcciones y sugerencias por parte del director de TFM y la comunicación entre ambas partes.

- **Desarrollo de módulos básicos:** Se han creado las diferentes funciones necesarias para implementar el módulo para la búsqueda de datos. Estos módulos básicos se han concebido como

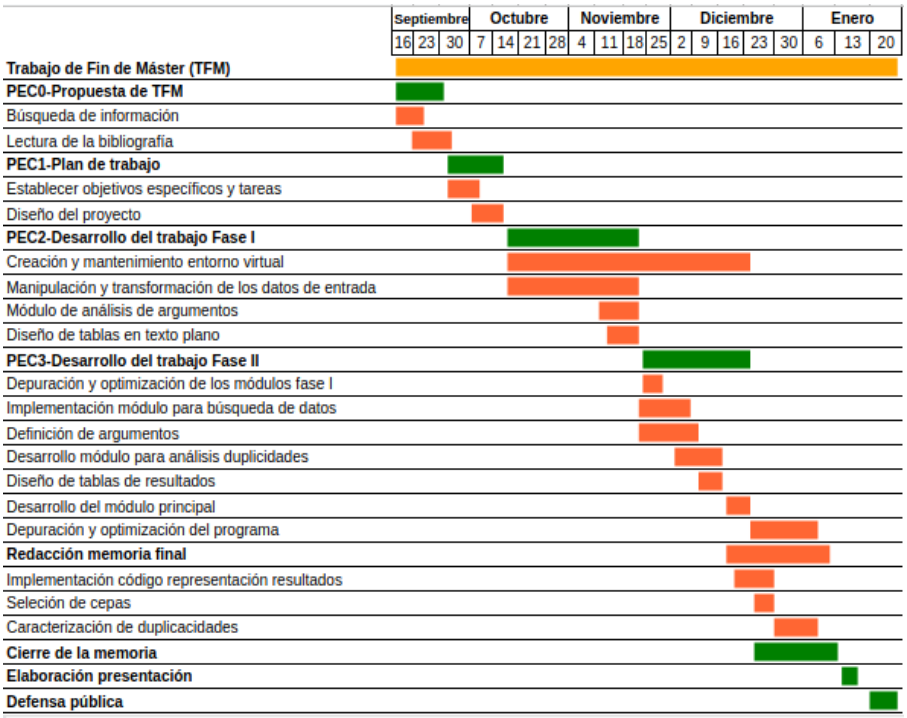


Figura 1.1: Cronograma detallado de la planificación del trabajo a lo largo de desarrollo del TFM. En verde se representan las entregas de documentación y en naranja las diferentes tareas realizadas.

pequeños fragmentos de código funcionales capaces de realizar las diferentes acciones de manipulación y transformación de los datos proporcionados. De esta forma, se obtendrán objetos que puedan ser utilizados en el módulo principal. Se han utilizado herramientas en BioPython profundizando en la documentación proporcionada para cada función y adaptándolas a nuestras necesidades para la creación de los nuevos programas.

- **Transformación de la información a texto plano:** mediante los programas Pandas y Numpy, se han manipulado los archivos de anotación genómicos (e.g. genbank, GTF, GFF) para crear objetos de python con la estructura adecuada para generar tablas y poder volcarlas como texto plano y visualizarlas, por ejemplo, en un procesador de hojas de cálculo. Estas tablas se han utilizado posteriormente para analizar las duplicidades génicas de los genomas estudiados.

- **Depuración de los módulos creados:** durante la primera fase del proyecto, se generaron una serie de módulos que debían ser depurados para su correcto funcionamiento. El seguimiento de errores de programación y su resolución han llevado un tiempo necesario para que el funcionamiento de los siguientes módulos asociados fuera correcto. Se ha implementado el programa de forma que sea muy flexible en cuanto al tipo de información proporcionada por el usuario.

- **Definición de argumentos:** para desarrollar una herramienta versátil y de fácil manejo por parte del usuario, se debe ofrecer un amplio rango de posibilidades que faciliten su uso sea cuál sea el tipo de archivo que se utilice para la búsqueda de duplicidades. Así, se han previsto

distintos escenarios para definir los argumentos posibles y generar los resultados en una única línea de comandos en la consola LINUX.

- **Diseño de tablas de resultados:** mediante los programas Pandas y Numpy se han definido los archivos de texto plano para formatear las tablas con los resultados obtenidos.
- **Desarrollo del módulo principal:** Para posibilitar la búsqueda de duplicidades en solo una línea de comando en la terminal, todos los módulos programados previamente deben relacionarse entre sí para poder funcionar como un conjunto. Este módulo principal recogerá los argumentos proporcionados por el usuario y, en base a ello, utilizará unas funciones u otras para generar el archivo de resultados.
- **Implementación del código para la representación de resultados:** Implementación del módulo en R que permitirá representar los resultados. Con el paquete BioCircos se ha diseñado un script adaptado a las peculiaridades de los datos generados por el programa en python para generar gráficos representativos del genoma estudiado y los diferentes grupos de duplicados relacionados entre sí.

## 1.5. Breve sumario de productos obtenidos

Durante el periodo de realización del TFM se han entregado los siguientes documentos:

- Propuesta de TFM.
- Pruebas de evaluación continua:
  - Definición de los contenidos del trabajo.
  - Plan de trabajo.
  - Informes de seguimiento

A estos documentos, se deben añadir la presente memoria y la presentación y defensa del TFM. Finalmente, se realizará un informe de autoevaluación que también será entregado con el resto de documentación.

Además de estos documentos, se han obtenido los siguientes productos:

- Secuencias de comandos (scripts) de los módulos en python.
- Script del programa en R.
- Tablas de anotación y gráficos de las duplicidades génicas de las cepas seleccionadas.
- Repositorio público en github con todos los scripts desarrollados, documentación y resultados generados. https://github.com/albamgarces/TFM_UOC_AMoya.git

## 1.6. Breve descripción de los otros capítulos de la memoria

El capítulo 2 se ha dedicado a explicar detalladamente cada módulo desarrollado y su funcionamiento. Cada una de las secciones del capítulo corresponde a las diferentes fases del proceso, desde la entrada de datos hasta la generación de resultados y su representación gráfica.

El capítulo 3 describe el uso de las herramientas creadas para el análisis de duplicidades en cepas bacterianas seleccionadas.

El apédice A contiene información sobre los requerimientos necesarios para utilizar los módulos implementados. Se resume brevemente los programas complementarios que se deben instalar y los enlaces al repositorio en github para la descarga de los scripts completos.

Por último, el apéndice B incluye los resultados obtenidos en el análisis de las cepas seleccionadas.

# Capítulo 2

# Desarrollo de herramientas bioinformáticas para el análisis de duplicidades genéticas

Para poder realizar una búsqueda de genes duplicados contenidos en un genoma se requiere obtener la información tanto de la anotación como de la secuencia proteica de cada gen codificante. Para ello es necesario un preprocesado de la información contenida en los diferentes ficheros de anotación y una posterior búsqueda de secuencias duplicadas. Posteriormente se representa toda esta información generada en un gráfico que muestre las relaciones entre las duplicaciones encontradas y su situación dentro del genoma.

A continuación se detalla cada fase del proceso y el desarrollo de las herramientas creadas para realizar cada tarea. Todo el código generado y la documentación necesaria para replicar el trabajo hecho se puede encontrar en la siguiente carpeta de github:

https://github.com/albamgarces/TFM_UOC_AMoya/tree/main/memoria

En la figura 2.1 se muestran los pasos del proceso de análisis desde la entrada de información hasta la generación resultados y gráficos.

## 2.1. Módulos en *python*

### 2.1.1. Requerimientos

El objetivo de este proyecto es comenzar el desarrollo de un paquete de python que permita la generación de análisis de duplicaciones a partir de ficheros de anotación. Para el desarrollo y testado de este paquete se ha utilizado la versión 3.8.5 de python y creado un entorno virtual específico para el TFM con el siguiente comando:

```
$ mkdir environments
$ cd environments
```

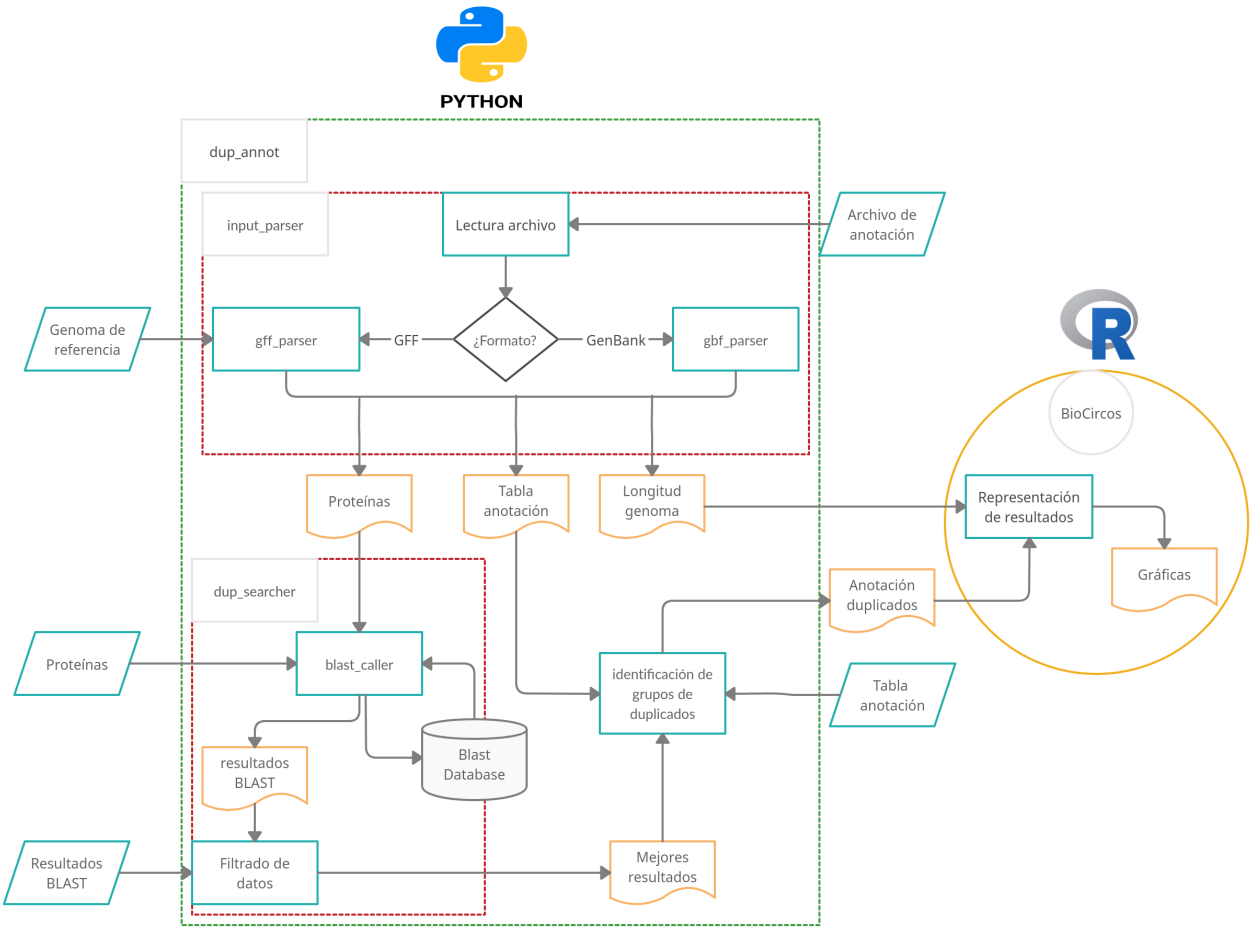


Figura 2.1: Diagrama de flujo de datos de los programas desarrollados donde puede seguirse el recorrido de los datos desde que entran en el programa hasta que se generan los resultados. Los paralelogramos corresponden a entrada de información desde el exterior (usuario); los rectángulos identifican cada función en python que desarrolla el proceso; las líneas de puntos engloban los módulos que contienen las correspondientes funciones en python; los círculos engloban los procesos desarrollados en R.

```
$ python3 -m venv TFM
```

Seguidamente, se instala el paquete de instalación *pip* que nos permitirá descargar módulos específicos necesarios para el funcionamiento de nuestro programa.

```
$ sudo apt install -y python3-pip
```

En el apéndice A se facilita la lista de módulos instalados a lo largo del desarrollo del programa y una breve descripción de los módulos instalados con *pip*.

Para la generación de búsquedas de similitud de secuencias, necesitamos hacer uso de BLAST+, en concreto de la herramienta Blastp, la cual compara secuencias de proteína a estudio con una base de datos de proteínas de referencia [23, 1]. En el apéndice A se muestra su instalación y funcionamiento desde el terminal [19].

### 2.1.2. Entrada de datos e información

La información proporcionada por el usuario puede ser de diferentes tipos y presentarse en diferentes formatos:

- **Anotación y ensamblaje de un genoma:** Pueden provenir de una base de datos RefSeq o GenBank como la de *National Center for Biotechnology Information* (NCBI) [18] o haber sido ensambladas *de novo*. Nuestro programa admitirá dos tipos de formatos que incluyan la anotación del genoma: GenBank y GFF3, este último deberá ir acompañado de un archivo fasta que contenga la secuencia genómica de referencia correspondiente mientras que los archivos en formato GenBank, ya tienen incluida esta información al final del documento.
- **Proteínas de interés:** Archivo en formato fasta con las proteínas traducidas. Pueden provenir de trabajos previos. Deben ir acompañadas de un archivo *csv* con la anotación del genoma.
- **Tabla de anotación:** Archivo en formato *csv* en el que aparezca la anotación del genoma a estudio. Debe ir acompañado de un archivo fasta con las secuencias proteicas.

Dependiendo del tipo de información, el tratamiento de la misma será diferente.

### 2.1.3. Tratamiento de datos de anotación: input_parser.py

Como se puede apreciar en la figura 2.2, el módulo input_parser.py se define con una serie de argumentos necesarios para activar el proceso. En caso de no proporcionar aquellos obligatorios (archivo de anotación y carpeta de salida donde se guarden los archivos generados) se generará un error que no permitirá continuar (Figura 2.3).

```
usage: inputParser [-h] -a  -o  [-r] [--debug]

Get proteins sequences from annotation file

optional arguments:
  -h, --help          show this help message and exit
  -r , --ref_file     Genome references FASTA file
  --debug

required named arguments:
  -a , --annot_file   Annotation file: genbank or GFF
  -o , --out_folder   Results folder
```

Figura 2.2: Información del uso de argumentos de input_parser.

Este módulo se encarga. en primer lugar, de tomar los ficheros de entrada y determinar su formato. En función de la extensión del archivo, llamará a la función que analiza archivos GFF o GenBank. Si la extensión no se incluye dentro de un listado determinado, se generará un error y el programa no continuará el proceso.

Puesto que la información contenida en cada tipo de fichero de anotación esta estructurada de forma diferente el manejo de esta debe ser acorde a sus características. Si tenemos un archivo GenBank, se llamará a **gbf_parser.py** para que lleve a cabo la conversión de los datos a las secuencias de proteínas traducidas. Además, generará un fichero de texto plano con la información contenida en el archivo de

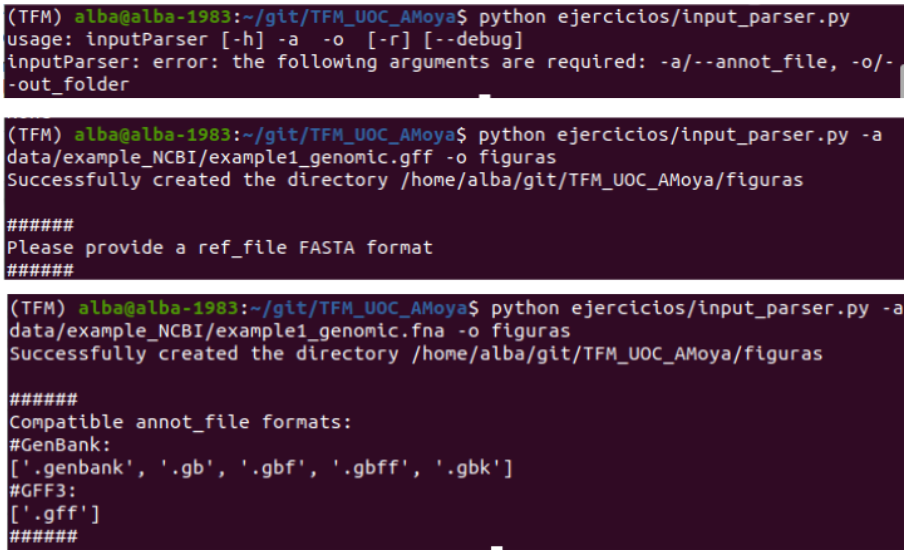


Figura 2.3: Diferentes errores que pueden ocurrir: Falta de carpeta de salida (arriba) o de archivo de referencia genómica (centro) y formato incorrecto (abajo).

anotación en forma de tabla. Si, en cambio, tenemos un archivo GFF, se invocará a **gff_parser.py**, el cual necesitará que también se facilite el archivo de referencia genómica para poder producir el archivo de proteínas. Si este segundo archivo no se proporciona, el programa detectará un error y no continuará el proceso (Figura 2.3).

Al terminar el proceso, obtendremos una carpeta de salida con tres documentos:

- df.csv: tabla de anotación del genoma que se utilizará (Figura 2.4). Está formada por 13 columnas de anotación que describen a cada una de las id únicas (protID) generadas por el programa para cada entrada.

| | locus_tag | protein_id | gene | start | end | strand | product | EC_number | old_locus_tag | inference |
|---|---|---|---|---|---|---|---|---|---|---|
| CDS_NC_009641.1_516_1878_pos | NWMN_RS00005 | A | dnaA | 516 | 1878 | pos | chromosomal replication initiator protein DnaA | | NWMN_0001 | COORDINATES: simil |
| CDS_NC_009641.1_2162_3296_pos | NWMN_RS00010 | B | dnaN | 2162 | 3296 | pos | DNA polymerase III subunit beta | 2.7.7.7 | NWMN_0002 | COORDINATES: simil |
| CDS_NC_009641.1_3676_3922_pos | NWMN_RS00015 | C | yaaA | 3676 | 3922 | pos | S4 domain-containing protein YaaA | | | COORDINATES: simil |
| CDS_NC_009641.1_3918_5031_pos | NWMN_RS00020 | D | recF | 3918 | 5031 | pos | DNA replication/repair protein RecF | | NWMN_0003 | COORDINATES: simil |
| CDS_NC_009641.1_5040_6975_pos | NWMN_RS00025 | E | | 5040 | 6975 | pos | DNA topoisomerase (ATP-hydrolyzing) subunit B | 5.6.2.2 | NWMN_0004 | COORDINATES: simil |
| CDS_NC_009641.1_7011_9675_pos | NWMN_RS00030 | F | gyrA | 7011 | 9675 | pos | DNA gyrase subunit A | 5.6.2.2 | NWMN_0005 | COORDINATES: simil |
| CDS_NC_009641.1_9761_10574_neg | NWMN_RS00035 | G | | 9761 | 10574 | neg | NAD(P)H-hydrate dehydratase | | NWMN_0006 | COORDINATES: simil |
| CDS_NC_009641.1_10899_12414_pos | NWMN_RS00040 | H | hutH | 10899 | 12414 | pos | histidine ammonia-lyase | 4.3.1.3 | NWMN_0007 | COORDINATES: simil |
| CDS_NC_009641.1_12792_14079_pos | NWMN_RS00045 | I | serS | 12792 | 14079 | pos | serine--tRNA ligase | 6.1.1.11 | NWMN_0008 | COORDINATES: simil |
| CDS_NC_009641.1_14728_15424_pos | NWMN_RS00050 | J | | 14728 | 15424 | pos | AzlC family ABC transporter permease | | NWMN_0009 | COORDINATES: simil |
| CDS_NC_009641.1_15420_15750_pos | NWMN_RS00055 | K | | 15420 | 15750 | pos | AzlD domain-containing protein | | NWMN_0010 | COORDINATES: simil |
| CDS_NC_009641.1_16112_17081_pos | NWMN_RS00060 | L | | 16112 | 17081 | pos | alpha/beta fold hydrolase | | NWMN_0011 | COORDINATES: simil |
| CDS_NC_009641.1_17371_18310_pos | NWMN_RS00065 | M | | 17371 | 18310 | pos | YybS family protein | | NWMN_0012 | COORDINATES: simil |
| CDS_NC_009641.1_18324_20292_pos | NWMN_RS00070 | N | gdpP | 18324 | 20292 | pos | cyclic-di-AMP phosphodiesterase GdpP | | NWMN_0013 | COORDINATES: simil |
| CDS_NC_009641.1_20288_20735_pos | NWMN_RS00075 | O | | 20288 | 20735 | pos | 50S ribosomal protein L9 | | NWMN_0014 | COORDINATES: simil |
| CDS_NC_009641.1_20766_22167_pos | NWMN_RS00080 | P | dnaB | 20766 | 22167 | pos | replicative DNA helicase | 3.6.4.12 | NWMN_0015 | COORDINATES: simil |

Figura 2.4: Tabla de anotación generada tras el tratamiento de los datos con input_parser.py.

- proteins.fa: archivo en formato fasta con las secuencias proteicas provenientes de la traducción de las diferentes regiones de codificación del genoma (CDS)

- length.csv: tabla en la que se incluye los diferentes identificadores de secuencia contenidos en los archivos de anotación proporcionados y su tamaño en pares de bases (pb).

### 2.1.4. Comparación de secuencias con BLAST+: blast_caller.py

BLAST+ (*Basic Local Alignment Search Tool*) es una herramienta informática de alineamiento de secuencias. Puede realizar búsquedas en las bases de datos de secuencias en cuestión de segundos para encontrar similitudes con las secuencias problema alineándolas por pares. Hay cinco variantes del algoritmo dependiendo del tipo de secuencias que se comparen entre sí: blastp para comparaciones entre proteínas; blastn compara secuecias de nucleótidos; blastx y y tblastn entre las posibles traducciones de secuencias de nucleótidos y bases de datos de proteínas y tblastx compara las seis traducciones posibles en sus marcos de lectura de la secuencia problema contra la seis de las secuencias de la base de datos de nucleótidos [21, 1].

Como se ha comentado al inicio del capítulo, utilizaremos la herramienta Blastp a nivel local. Es decir, invocaremos al algoritmo con los parámetros adecuados para que utilice como bases de datos las mismas secuencias de proteínas con las que después tratará de encontrar los mejores alineamientos. Cada uno de estos alineamientos obtendrá una puntuación de calidad que indicarán el grado de similitud de las dos secuencias comparadas [21, 1].

El módulo blast_caller.py se divide en dos fases:

- **Generación de la base de datos** local a partir del archivo de proteínas generado anteriormente.
- **Generación resultados BLAST** comparando las secuencias de la base de datos con el mismo archivo de proteínas (las secuencias se comparan entre sí mismas dos a dos).

Este módulo se define con los argumentos requeridos y opcionales mostrados en la figura 2.5. Es necesario que el usuario aporte un archivo fasta de proteínas para que el programa funcione. Además, se debe añadir la ruta donde se encuentran los archivos ejecutables de BLAST para que puedan ser utilizados.

```
usage: blastCaller [-h] [-d] -b  -f  [-o] [--debug]

Create a BLAST database

optional arguments:
  -h, --help            show this help message and exit
  -d , --db_name        New database name
  -o , --out_folder     Results folder
  --debug

required named arguments:
  -b , --blast_folder   BLAST binary folder
  -f , --fasta_file     Protein sequences FASTA file
```

Figura 2.5: Información del uso de argumentos de blast_caller

El programa permite generar el comando necesario para ejecutar BLAST+ haciendo una llamada al sistema utilizando la funcion *system* de python (Figura 2.6).

Al finalizar el proceso, por un lado, se obtienen tres archivos proteins_db que forman la base de datos de proteínas, donde las extensiones *phr*, *pin* y *psq* corresponden a las cabeceras, índices y secuencias, respectivamente.

```
(TFM) alba@alba-1983:~/git/TFM_UOC_AMoya$ python ejercicios/blast_caller.py -f
 input_parser_files/proteins.fa -o blast_caller_files -b /usr/bin

[** System: /usr/bin/makeblastdb -in input_parser_files/proteins.fa -input_typ
e fasta -dbtype prot -out /home/alba/git/TFM_UOC_AMoya/blast_caller_files/prot
eins_db **]

[** System: /usr/bin/blastp -query input_parser_files/proteins.fa -db /home/al
ba/git/TFM_UOC_AMoya/blast_caller_files/proteins_db -outfmt '6 std qlen slen'
-num_threads 1 -out /home/alba/git/TFM_UOC_AMoya/blast_caller_files/proteins_B
LAST_raw_results.txt **]
```

Figura 2.6: Linea de comandos en la terminal para ejecutar makeblastdb y blastp

Por otro lado, obtenemos un fichero de con los resultados BLAST en bruto de las secuencias similares obtenidas (hits). Este último documento se presenta como una tabla con las siguientes columnas [24, 25]:

- id de la secuencia problema
- id de la secuencia de referencia
- porcentaje de coincidencias idénticas
- longitud del alineamiento
- número de no concordancias
- número de huecos
- comienzo del alineamiento en la secuencia problema
- final del alineamiento en la secuencia problema
- comienzo del alineamiento en la secuencia de referencia
- comienzo del alineamiento en la secuencia de referencia
- e-valor: número de hits de igual calidad que se espera encontrar debido al azar. Los resultados aparecen ordenados por defecto según su e-valor, los mejores hits (valores más bajos) aparecen primero. Por defecto, se establece en un valor de 10, por encima del cual los alineamientos no se incluyen en el documento.
- bit-score: tamaño de una base de datos requerido para encontrar el mismo número de concordancia por azar. Las secuencias presentan mayor similitud cuanto mayor sea su bit-score.
- longitud de la secuencia problema
- longitud de la secuencia de referencia

### 2.1.5. Búsqueda de duplicados: dup_searcher.py

Este módulo ofrece dos posibilidades de datos de entrada. El usuario puede proporcionar el archivo fasta de proteínas, que hará que el módulo conecte con blast_caller para generar el documento de resultados. O bien, puede proporcionar directamente un documento de resultados BLAST creado previamente.

Los argumentos definidos para dup_searcher (figura 2.7) nos permite cambiar las variables de filtrado de los datos para generar unos resultados más o menos afinados. Así, podemos cambiar los valores mínimos de bit-score, e-valor o los porcentajes de alineamiento o similitud, por debajo de los

cuales serán eliminados de la lista de resultados. Paralelamente, se filtran las parejas espejo (cuando A=B y B=A) y las que correspondan a una secuencia alineada consigo misma (A=A). Por defecto, se ha diseñado el programa para trabajar con un e-valor = $10^{-05}$ (un e-valor bajo nos permitirá tener pocos resultados pero de buena calidad), bit-score = 50 y unos porcentajes de alineamiento y similitud del 85 %.

```
usage: dupSearcher [-h] [-t] [-f] [-o] [--debug] [-b] [-bs] [-d] [-e] [-p] [-pi]

Search for genomic duplicated proteins

optional arguments:
  -h, --help            show this help message and exit
  -t , --text_file      Blast raw results text file
  -f , --fasta_file     Protein sequences FASTA file
  -o , --out_folder     Results folder
  --debug

BLAST options named arguments:
  -b , --blast_folder   BLAST binary folder. **Note blast_folder=/usr/bin is set by
                        default**
  -bs , --bitscore      BLAST bit-score: requires size of a sequence database in which
                        the current match could be found just by chance. **Note
                        bit_score = 50 is set by default**
  -d , --db_name        New database name
  -e , --evalue         BLAST e-value: number of expected hits of similar quality
                        (score) that could be found just by chance. **Note e-value =
                        1e-05 is set by default**
  -p , --percentage     Percentage of alignment in query. *Note pident = 85 is set by
                        default**
  -pi , --pident        Percentage of similarity in alignment. **Note percentage = 85
                        is set by default**
```

Figura 2.7: Información del uso de argumentos de dup_searcher

El resultado final es una tabla ordenada según el porcentaje de alineamiento en primer lugar, seguido por un orden ascendente de e-valor (los mejores resultados, e-valor menor, primero) y bitscore en sentido descendente. Añadimos los encabezados de las columnas y obtenemos un documento similar a la figura 2.8

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | dup_id | rec_id | locus_tag | protein_id | gene | start | end | strand | pseudo | product | db_x | EC_n | old_locu | inference |
| 2 | CDS_CP065453.1_124800_124986_neg | 111 | CP065453.1 | I5M60_00650 | QPP59745.1 | | 124800 | 124986 | neg | | general stress protein | | | | COORDINATES: similar to A |
| 3 | CDS_CP065453.1_1328563_1328995_pos | 83 | CP065453.1 | I5M60_06490 | QPP60799.1 | | 1328563 | 1328995 | pos | | Arc family DNA-binding protein | | | | COORDINATES: similar to A |
| 4 | CDS_CP065453.1_878235_879339_pos | 32 | CP065453.1 | I5M60_04410 | QPP60415.1 | | 878235 | 879339 | pos | | phosphotransferase | | | | COORDINATES: similar to A |
| 5 | CDS_CP065453.1_4622460_4622784_pos | 91 | CP065453.1 | I5M60_22630 | QPP58911.1 | | 4622460 | 4622784 | pos | | DUF968 domain-containing protein | | | | COORDINATES: similar to A |
| 6 | CDS_CP065455.1_57622_58828_pos | 26 | CP065455.1 | I5M60_27955 | QPP62612.1 | | 57622 | 58828 | pos | | IS4 family transposase | | | | COORDINATES: similar to A |
| 7 | CDS_CP065453.1_329092_330514_pos | 15 | CP065453.1 | I5M60_01705 | QPP59942.1 | | 329092 | 330514 | pos | | FAD-binding oxidoreductase | | | | COORDINATES: similar to A |
| 8 | CDS_CP065453.1_562740_563634_neg | 55 | CP065453.1 | I5M60_02870 | QPP60146.1 | | 562740 | 563634 | neg | | SUMF1/EgtB/PvdO family nonheme iron enzyme | | | | COORDINATES: similar to A |
| 9 | CDS_CP065455.1_0_1099_pos | 33 | CP065455.1 | I5M60_27575 | QPP62533.1 | | 0 | 1099 | pos | | TonB-dependent receptor | | | | COORDINATES: similar to A |
| 10 | CDS_CP065453.1_4334691_4335417_pos | 57 | CP065453.1 | I5M60_21270 | QPP58658.1 | | 4334691 | 4335417 | pos | | histidine utilization repressor | | | | COORDINATES: similar to A |
| 11 | CDS_CP065453.1_4614588_4614747_pos | 112 | CP065453.1 | I5M60_22555 | QPP62289.1 | | 4614588 | 4614747 | pos | | adenylate cyclase | | | | COORDINATES: similar to A |
| 12 | CDS_CP065453.1_5103440_5104229_pos | 56 | CP065453.1 | I5M60_25150 | QPP59368.1 | paaB | 5103440 | 5104229 | pos | | 2-(1,2-epoxy-1,2-dihydrophenyl)acetyl-CoA isomerase | | | | COORDINATES: similar to A |
| 13 | CDS_CP065453.1_4446997_4447213_pos | 104 | CP065453.1 | I5M60_21785 | QPP58755.1 | | 4446997 | 4447213 | pos | | hypothetical protein | | | | COORDINATES: similar to A |
| 14 | CDS_CP065453.1_561663_562557_neg | 55 | CP065453.1 | I5M60_02860 | QPP60145.1 | | 561663 | 562557 | neg | | SUMF1/EgtB/PvdO family nonheme iron enzyme | | | | COORDINATES: similar to A |
| 15 | CDS_CP065453.1_4440734_4441760_neg | 126 | CP065453.1 | I5M60_21750 | QPP58748.1 | | 4440734 | 4441760 | neg | | phage portal protein | | | | COORDINATES: similar to A |
| 16 | CDS_CP065456.1_1055_1931_neg | 49 | CP065456.1 | I5M60_28035 | QPP62614.1 | | 1055 | 1931 | neg | | class A extended-spectrum beta-lactamase CTX-M-14 | | | | COORDINATES: similar to A |
| 17 | CDS_CP065453.1_4458998_4459514_pos | 78 | CP065453.1 | I5M60_21875 | QPP58771.1 | | 4458998 | 4459514 | pos | | phage major tail tube protein | | | | COORDINATES: similar to A |
| 18 | CDS_CP065453.1_4447193_4447703_pos | 80 | CP065453.1 | I5M60_21790 | QPP58756.1 | | 4447193 | 4447703 | pos | | lysozyme | | | | COORDINATES: similar to A |
| 19 | CDS_CP065454.1_119282_120251_neg | 37 | CP065454.1 | I5M60_27210 | QPP62462.1 | | 119282 | 120251 | neg | | IS5-like element IS903B family transposase | | | | COORDINATES: similar to A |
| 20 | CDS_CP065453.1_4624676_4625129_pos | 81 | CP065453.1 | I5M60_22660 | QPP58916.1 | | 4624676 | 4625129 | pos | | DNA-packaging protein | | | | COORDINATES: similar to A |
| 21 | CDS_CP065453.1_1492384_1492957_neg | 73 | CP065453.1 | I5M60_07330 | QPP60955.1 | | 1492384 | 1492957 | neg | | phage baseplate assembly protein V | | | | COORDINATES: similar to A |

Figura 2.8: Tabla de anotación generada tras el tratamiento de los datos con input_parser.py

### 2.1.6. Anotación de las proteínas duplicadas: dup_annot.py

Este módulo engloba a todos los anteriores. Los argumentos definidos (figura 2.9) lo dotan de la flexibilidad necesaria para generar los resultados independientemente del tipo de información proporcionada, iniciando el proceso a partir del punto correspondiente según la misma.

Además de los argumentos incluidos en los módulos anteriores, añadimos uno nuevo que permitirá elegir si se quiere tener en cuenta los pseudogenes o no en el resultado final.

```
usage: dupAnnotation [-h] [-a] [-r] [-b] [-bs] [-d] [-e] [-p] [-pi] [-c] [-t]
                     [--pseudo] [-o] [--debug]

Get an annotation file with duplicated protein on genome.

optional arguments:
  -h, --help            show this help message and exit
  -c , --annot_table    Genome annotation .csv file previously analyzed.
  -t , --text_file      Blast raw results text file.
  --pseudo              Wether to use pseudogenes or not
  -o , --out_folder     Results folder
  --debug

Annotation parser options named arguments:
  -a , --annot_file     Annotation file: genbank or GFF.
  -r , --ref_file       Genome references FASTA file.

BLAST options named arguments:
  -b , --blast_folder   BLAST binary folder. **Note blast_folder=/usr/bin is set by
                        default**
  -bs , --bitscore      BLAST bit-score: requires size of a sequence database in which
                        the current match could be found just by chance. **Note
                        bit_score = 50 is set by default**
  -d , --db_name        New database name
  -e , --evalue         BLAST e-value: number of expected hits of similar quality
                        (score) that could be found just by chance. **Note e-value =
                        1e-05 is set by default**
  -p , --percentage     Percentage of alignment in query. *Note pident = 85 is set by
                        default**
  -pi , --pident        Percentage of similarity in alignment. **Note percentage = 85
                        is set by default**
```

Figura 2.9: Información del uso de argumentos de dup_annot

El módulo dup_annot.py toma los resultados en bruto de BLAST e identifica los diferentes grupos de duplicados, teniendo en cuenta que si A=B y B=C, entonces A=B=C. Esta nueva información la volcará en la tabla de anotación generada en input_parser (o proporcionada por el usuario), de la cual se mantendrán únicamente las proteínas duplicadas. De esta forma, obtenemos una tabla de anotación completa para cada una de las proteínas duplicadas, identificadas por el grupo de duplicado al que pertenecen.

Al finalizar el proceso, el programa muestra en pantalla cuántos grupos y proteínas duplicadas ha encontrado con los parámetros de BLAST introducidos (figura 2.10)

### 2.1.7. Archivos obtenidos

Como muestra la figura (2.11) al final del proceso obtenemos una carpeta con todos los archivos generados juntos.

- Del tratamiento de datos con input_parser se obtienen los archivos df.csv (tabla de anotación completa), proteins.fa (secuencias de proteínas) y length.csv (longitudes de los diferentes genomas contenidos).

- La búsqueda de resultados con dup_searcher nos proporciona los archivos pertenecientes a la base de datos creada (proteins_db), el archivo proteins_BLAST_raw_resuts.txt con los alineamientos generados y filtered_results.csv tras pasar por el proceso de filtrado de los mejores resultados.

- Finalmente, dup_annot.csv es la tabla de anotación para las proteínas identificadas como duplicados.

```
(TFM) alba@alba-1983:~/git/TFM_UOC_AMoya$ python ejercicios/dup_annot.py -a data
/example_NCBI/example1_genomic.gbff -o dup_annot_files
######
e-value = 1e-05
bitscore = 50
percentage alignment in query = 85
percentage of identity = 85
blast_folder = /usr/bin
######
Successfully created the directory /home/alba/git/TFM_UOC_AMoya/dup_annot_files


[** System: /usr/bin/makeblastdb -in /home/alba/git/TFM_UOC_AMoya/dup_annot_file
s/proteins.fa -input_type fasta -dbtype prot -out /home/alba/git/TFM_UOC_AMoya/d
up_annot_files/proteins_db **]

[** System: /usr/bin/blastp -query /home/alba/git/TFM_UOC_AMoya/dup_annot_files/
proteins.fa -db /home/alba/git/TFM_UOC_AMoya/dup_annot_files/proteins_db -outfmt
 '6 std qlen slen' -num_threads 1 -out /home/alba/git/TFM_UOC_AMoya/dup_annot_fi
les/proteins_BLAST_raw_results.txt **]
#####
Found 133 groups of duplicates with a total of 238 proteins duplicated from 2914
 proteins on the original file
#####
```

Figura 2.10: Mensaje mostrado en la terminal de Linux al terminar el proceso de dup_annot.py.

## 2.2. Módulo en ℝ

Se ha utilizado la versión 4.0.3 de R, en el entorno de desarrollo RStudio (versión 1.3.1073). Se ha creado un proyecto específico para desarrollar el paquete cuyas especificaciones se detallan en el apéndice A.

Se ha utilizado el paquete **BioCircos** [12, 29] para generar gráficos circulares que representen el cromosoma bacteriano y sus posibles plásmidos, resaltando los genes duplicados y sus grupos localizados en la primera fase del proyecto. En la figura 2.12 se puede ver un ejemplo de gráfico generado con BioCircos.

A continuación se hará una breve descripción de las funciones más significativas del módulo.

- **duplicate_group_tracks:** para cada grupo de duplicados se determinará el cromosoma o plásmido a los que pertenecen sus componentes (rec_id), posición e inicio de cada secuencia. Se generará un *track* en el que las secuencias duplicadas se conectaran con líneas con los miembros de su grupo de duplicación.

- **parse_data**: genera la información que se mostrará al pasar el cursor del ratón por encima de cada secuencia duplicada y la paleta de colores que identificará cada tipo de información.

- **add_genes_strand:** para cada hebra del cromosoma, se creará un círculo o track donde se podrán ver los genes duplicados alojados.

- **create_BioCircos:** finalmente se crea el gráfico con la información generada en las funciones previas.

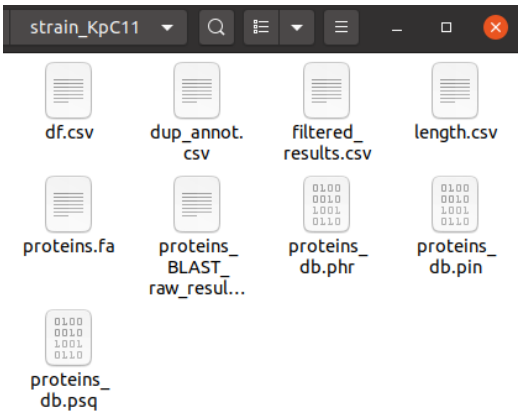


Figura 2.11: Carpeta de resultados generada tras ejecutar el programa dup_annot.py

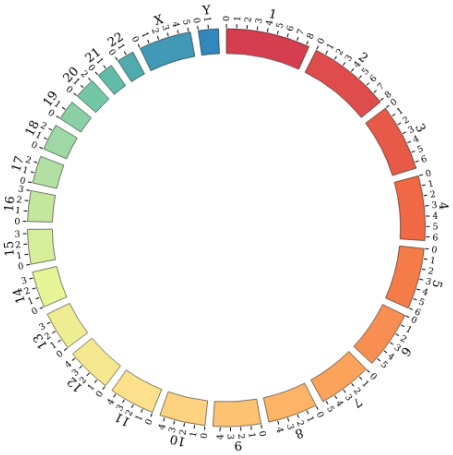

Figura 2.12: Gráfico generado automáticamente con los datos por defecto del paquete BioCircos de R. Se pueden ver los diferentes cromosomas del genoma humano identificados por colores y con un tamaño proporcional con respecto a la totalidad del genoma.

# Capítulo 3

# Análisis de duplicaciones génicas en una selección de cepas bacterianas

Con el fin de complementar la caracterización de duplicidades en cepas de *E. coli* y diferentes especies de estafilococos y enterococos descritas anteriormente [14, 15] y como ejemplo de uso de las herramientas desarrolladas durante este TFM descritas en el capítulo 2 se ha hecho un análisis de la duplicación génica en cepas bacterianas del grupo ESKAPE. Se han utilizado las especies del grupo no incluidas en estos trabajos (***Klebsiella pneumoniae, Acinetobacter baumannii, Pseudomonas aeruginosa, Enterobacter* spp.**.

Este grupo de bacterias se caracteriza por ser especialmente patógenas al presentar resistencia a los antibióticos conocidos para el tratamiento de las enfermedades derivadas de ellas [9, 16]. Son bacterias Gram-negativas de marcado comportamiento oportunista que colonizan las vías respiratorias y digestivas de individuos inmunodeprimidos provocando infecciones y neumonías graves. Al ser muy resistentes a los antibióticos conocidos su tratamiento puede complicarse llegando a la muerte del paciente [7, 10, 27, 20].

## 3.1. Selección de cepas

Para realizar la selección de cepas a analizar, se ha procedido a realizar una búsqueda en la base de datos de genomas del NCBI [22] para cada una de las especies a estudio. Se han filtrado los resultados para obtener genomas completos y patógenas para el ser humano, los resultados obtenidos se ordenaron por fecha de actualización y se eligieron las cuatro cepas más actuales de cada especie (figura 3.1). En la tabla 3.1 se muestran las cepas seleccionadas.

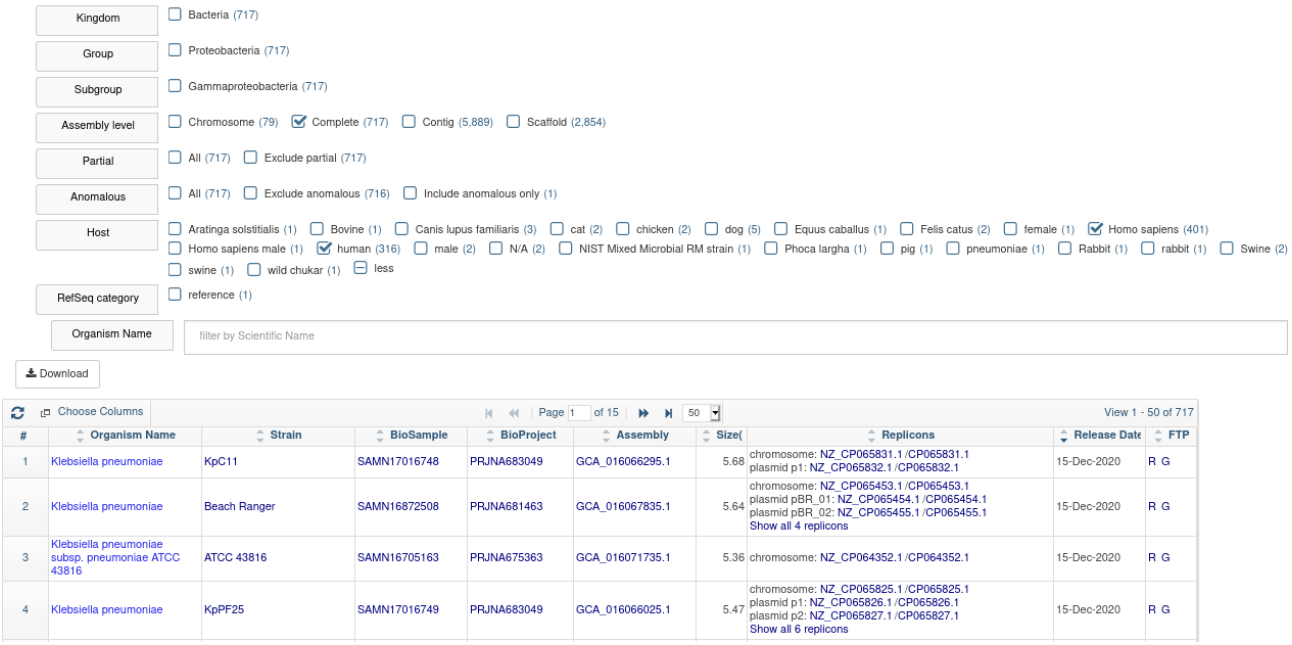


Figura 3.1: Resultados obtenidos en NCBI al hacer una búsqueda de la cepa de interés y aplicar los filtros deseados. Los resultados se han ordenado por fecha de actualizaciónn para poder seleccionar las cepas con los datos más actualizados.

| Especie | Cepa |
|---|---|
| *Klebsiella pneumoniae* | KpC11<br>ATCC 43816<br>Beach Ranger<br>KpPF25 |
| *Acinetobacter baumannii* | ATCC 17961<br>TP3<br>TP2<br>FDAARGOS_1036 |
| *Pseudomonas aeruginosa* | DL201330<br>TJ2014-049<br>TJ2019-017<br>TJ2019-022 |
| *Enterobacter cloacae* | STN0717-73<br>STN0717-60<br>RHBSTW-00399<br>RHBSTW-00490 |

Cuadro 3.1: Cepas seleccionadas para llevar a cabo la búsqueda de duplicidades y ejemplo de uso de la herramienta.

## 3.2. Búsqueda y representación de duplicidades

A continuación se muestra como ejemplo de uso de las herramientas desarrolladas en la búsqueda, anotación y representación de duplicidades génicas de la cepa Beach Ranger de *Klebsiella pneumoniae*. El motivo de la elección de esta cepa como muestra de ejemplo es meramente didáctico. Es un grupo de bacterias con varios plásmidos de diferentes tamaños y muestra duplicidades tanto en las hebras positivas como negativas. Estas características se prestan a generar unas gráficas visualmente atractivas y completas para explicar cada uno de los elementos que nos encontramos.

**- Datos:** Los archivos de anotación completos se pueden descargar directamente del servidor de búsqueda del NCBI, accediendo a la carpeta ftp del servidor (figura 3.2) desde el link que se ofrece en la columna correspondiente. Si se desea el genoma del cromosoma o un plásmido en concreto, se puede acceder a ellos directamente desde los enlaces de la tabla de resultados de búsqueda.

Índice de ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/016/067/835/GCA_016067835.1_ASM1606783v1/

Subir al directorio superior.

| Nombre | Tamaño | Última modificación |
|---|---|---|
| Archivo: GCA_016067835.1_ASM1606783v1_assembly_report.txt | 2 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_assembly_stats.txt | 7 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_cds_from_genomic.fna.gz | 1721 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_feature_count.txt.gz | 1 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_feature_table.txt.gz | 215 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_genomic.fna.gz | 1627 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_genomic.gbff.gz | 3901 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_genomic.gff.gz | 343 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_genomic.gtf.gz | 407 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_protein.faa.gz | 1066 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_protein.gpff.gz | 2761 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_rna_from_genomic.fna.gz | 7 KB | 19/12/20 21:06:00 CET |
| Archivo: GCA_016067835.1_ASM1606783v1_translated_cds.faa.gz | 1236 KB | 19/12/20 21:06:00 CET |
| *README.txt* | | 19/12/20 21:06:00 CET |
| Archivo: annotation_hashes.txt | 1 KB | 19/12/20 21:06:00 CET |
| Archivo: assembly_status.txt | 1 KB | 3/1/21 18:22:00 CET |
| Archivo: md5checksums.txt | 2 KB | 25/12/20 9:01:00 CET |

Figura 3.2: Contenido de archivos de la carpeta correspondiente a la cepa Beach Ranger del servidor ftp de NCBI

Hemos seleccionado la anotación en formato *GenBank* (extensión gbff) ya que nos permite utilizar un único documento, favoreciendo la agilidad del proceso.

**- Búsqueda y anotación de duplicados:** Siguiendo las indicaciones dadas en el capítulo anterior. Invocamos desde la terminal el módulo dup_annot.py y aportamos la información conveniente. Dejamos los valores de los parámetros que se dan por defecto y generamos los resultados (figura 3.3).

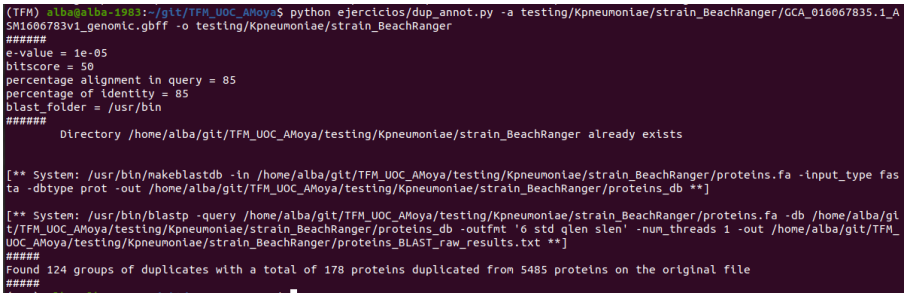


Figura 3.3: Resultados obtenidos en la terminal al invocar dup_annot.py con los datos para la cepa Beach Ranger.

**- Resultados obtenidos:** La figura 3.4 muestra un fragmento de la tabla de anotación para esas 178 proteínas duplicadas en la cepa Beach Ranger.

| | | dup_id | rec_id | locus_tag | protein_id | gene | start | end | strand | pseudo | product |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | CDS_CP065453.1_124800_124986_neg | 111 | CP065453.1 | I5M60_00650 | QPP59745.1 | | 124800 | 124986 | neg | | general stress protein |
| 3 | CDS_CP065453.1_1328563_1328995_pos | 83 | CP065453.1 | I5M60_06490 | QPP60799.1 | | 1328563 | 1328995 | pos | | Arc family DNA-binding protein |
| 4 | CDS_CP065453.1_878235_879339_pos | 32 | CP065453.1 | I5M60_04410 | QPP60415.1 | | 878235 | 879339 | pos | | phosphotransferase |
| 5 | CDS_CP065453.1_4622460_4622784_pos | 91 | CP065453.1 | I5M60_22630 | QPP58911.1 | | 4622460 | 4622784 | pos | | DUF968 domain-containing protein |
| 6 | CDS_CP065455.1_57622_58828_pos | 26 | CP065455.1 | I5M60_27955 | QPP62612.1 | | 57622 | 58828 | pos | | IS4 family transposase |
| 7 | CDS_CP065453.1_329092_330514_pos | 15 | CP065453.1 | I5M60_01705 | QPP59942.1 | | 329092 | 330514 | pos | | FAD-binding oxidoreductase |
| 8 | CDS_CP065453.1_562740_563634_neg | 55 | CP065453.1 | I5M60_02870 | QPP60146.1 | | 562740 | 563634 | neg | | SUMF1/EgtB/PvdO family nonheme iron enzyme |
| 9 | CDS_CP065455.1_0_1099_pos | 33 | CP065455.1 | I5M60_27575 | QPP62533.1 | | 0 | 1099 | pos | | TonB-dependent receptor |
| 10 | CDS_CP065453.1_4334691_4335417_pos | 57 | CP065453.1 | I5M60_21270 | QPP58658.1 | | 4334691 | 4335417 | pos | | histidine utilization repressor |
| 11 | CDS_CP065453.1_4614588_4614747_pos | 112 | CP065453.1 | I5M60_22555 | QPP62289.1 | | 4614588 | 4614747 | pos | | adenylate cyclase |
| 12 | CDS_CP065453.1_5103440_5104229_pos | 56 | CP065453.1 | I5M60_25150 | QPP59368.1 | paaB | 5103440 | 5104229 | pos | | 2-(1,2-epoxy-1,2-dihydrophenyl)acetyl-CoA isomerase |
| 13 | CDS_CP065453.1_4446997_4447213_pos | 104 | CP065453.1 | I5M60_21785 | QPP58755.1 | | 4446997 | 4447213 | pos | | hypothetical protein |
| 14 | CDS_CP065453.1_561663_562557_neg | 55 | CP065453.1 | I5M60_02860 | QPP60145.1 | | 561663 | 562557 | neg | | SUMF1/EgtB/PvdO family nonheme iron enzyme |
| 15 | CDS_CP065453.1_4440734_4441760_neg | 126 | CP065453.1 | I5M60_21750 | QPP58748.1 | | 4440734 | 4441760 | neg | | phage portal protein |

Figura 3.4: Tabla de anotación de las proteínas duplicadas generada tras el tratamiento de los datos de la cepa Beach Rangercon input_parser.py

En el apéndice B se puede encontrar los resultados obtenidos para la totalidad de las cepas estudiadas.

**- Representación gráfica:** Finamente, utilizando los comandos de R procedemos a generar la gráfica que nos mostrará la distribución de las proteínas duplicadas.

Le indicamos al programa cuáles son los datos que debe utilizar con las siguientes instrucciones:

```
## strain BeachRanger
seq_lengths_BeachRanger <- read.csv(paste0(data_folder, "Kpneumoniae/strain_
    BeachRanger/length.csv"),
                                    header=FALSE, row.names=1)
bed_info_file_BeachRanger <- read.table(paste0(data_folder, "Kpneumoniae/strain_
    BeachRanger/dup_annot.csv"), sep=",",
                                        header=TRUE)
bed_info_file_BeachRanger <- bed_info_file_BeachRanger[,c("dup_id", "rec_id", "start"
    , "end", "locus_tag", "product", "strand")]
```

Se debe considerar que el valor por defecto para los pseudogenes en dup_annot es no tenerlos en cuenta a la hora de generar la tabla final, pero sí que se incluyeron en los grupos de duplicados correspondiente. Esto puede resultar en que nos queden grupos de duplicados con una única proteína,

lo que nos generará problemas a la hora de intentar graficar los resultados. Para evitar esto, eliminamos esas proteínas huérfanas con el siguiente comando:

```
# # keep only real duplicated groups
bed_info_file_BeachRanger <- bed_info_file_BeachRanger %>% group_by(dup_id) %>%
    filter(n() >1)
#
```

Finalmente, creamos nuestro BioCircos (figura 3.5):

```
create_BioCircos(seq_lengths = seq_lengths_BeachRanger,
                 bed_info_file = bed_info_file_BeachRanger)
```

En el apéndice B se adjunta los cuatro gráficos correspondientes a cada cepa de *K pneumoniae* seleccionada con objetivo comparativo. También se aporta el enlace a la carpeta github correspondiente donde encontrar todas las gráficas de las cepas analizadas y los archivos generados tras su análisis.

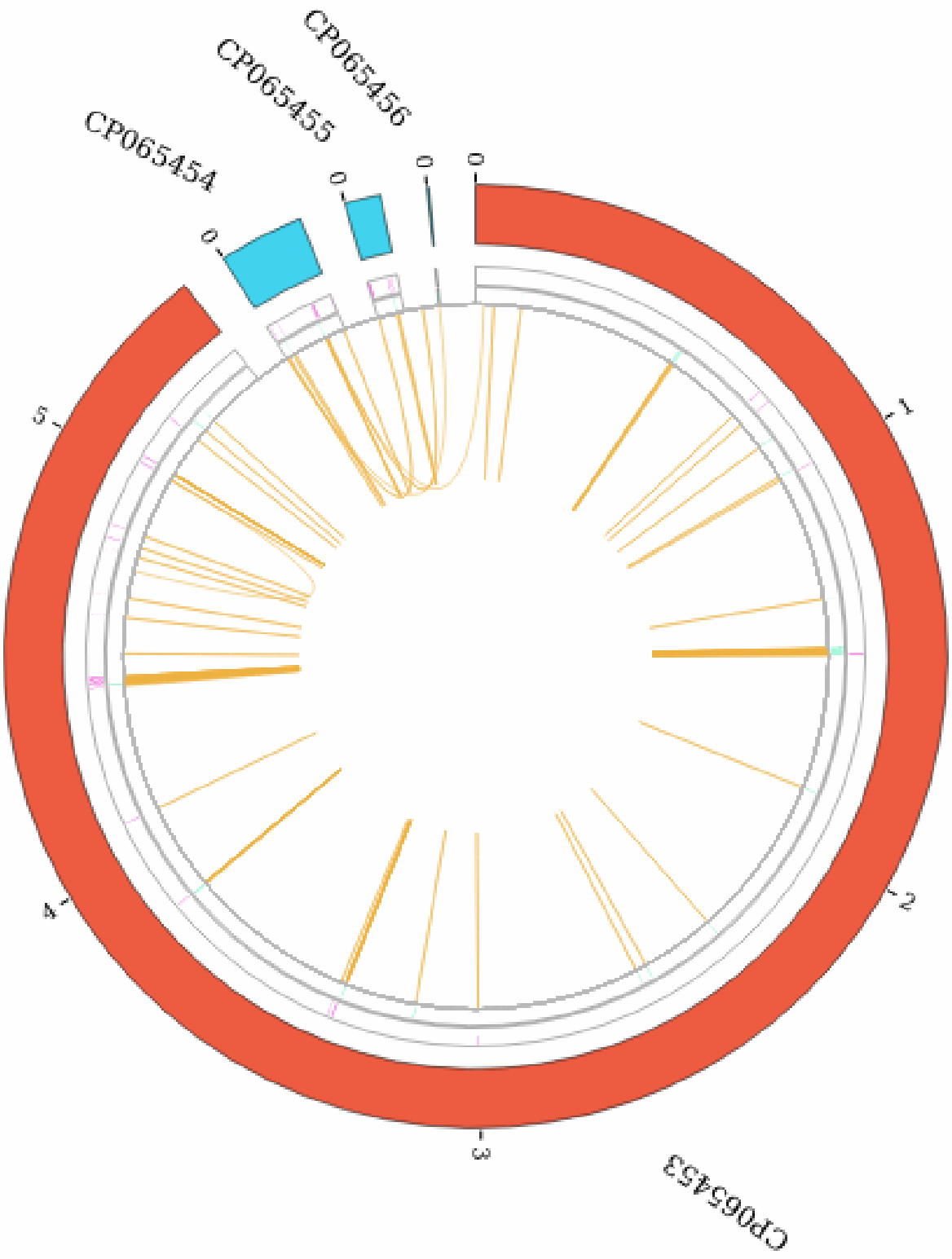


Figura 3.5: Representación gráfica con BioCircos de los genes duplicados en el genoma de la cepa Beach Ranger de *K. pneumoniae*. El cromosoma de la bacteria (rojo) y sus plásmidos (azul) se representan en un mapa circular que muestra la localización de cada gen duplicado. Las siguientes dos capas muestran los genes en la hebra positiva (rosa) y negativa (turquesa). Las líneas naranja interiores señalan las conexiones entre proteínas pertenecientes al mismo grupo de duplicados. Escala de tamaño en Mb

# Conclusiones

En este proyecto se ha diseñado una línea de trabajo para la búsqueda de duplicidades genómicas a partir de ficheros de anotación proporcionados por el usuario. Para llevar a cabo el proceso, se ha desarrollado una serie de programas o módulos en python y R que automatizan el análisis de los datos. El resultado final del proceso de análisis es una tabla de anotación de las proteínas duplicadas halladas en el genoma a estudio y una representación gráfica de las mismas.

El trabajo ha supuesto un gran reto tanto en su diseño como en la depuración del código y puesta a punto de los programas debido al gran esfuerzo que supone aprender un lenguaje de programación en pocos meses y resolver los problemas y errores del sistema que se iban dando.

No obstante, el desarrollo de este TFM ha permitido adquirir una serie de competencias que se resumen a continuación:

- Crear un plan de trabajo desde el inicio para automatizar la búsqueda y representación de duplicidades génicas. Se ha tenido que ir adaptando a los plazos previstos conforme han surgido imprevistos que ralentizaban el avance del trabajo. También se han tenido que tomar decisiones a la hora de establecer prioridades para la consecución de los objetivos o relajar la depuración de los códigos.

- Aprender a tratar la anotación genómica a partir de distintos formatos.

- Desarrollo del código para las diferentes funciones que llevan a cabo cada fase del proceso. Cada una de ellas debía ser autónoma y ser capaz de generar resultados por sí mismas.

- Diseño de una herramienta capaz de automatizar todo el proceso a partir de pequeñas funciones autónomas que debían ser relacionadas entre sí para trabajar como un todo. Desde el tratamiento de los datos iniciales hasta la representación de los resultados obtenidos. se ha tenido que considerar múltiples factores y variables que hicieran la herramienta final lo más versátil y universal posible. Esta fase ha sido especialmente complicada debido a que había que tener en cuenta diversos factores que podían provocar la pérdida de autonomía de las funciones individuales.

- Adaptar scripts de código preexistentes a las necesidades del TFM ajustándolos a las características específicas del proyecto.

- Escribir nuevo código y profundizar en la comprensión de las estructuras de los lenguajes de programación *python* y *R*. Lejos de lo que le podría parecer a alguna persona ajena al desarrollo

de código informático, cada lenguaje tiene sus propias reglas y estructuras y se debe aprender a diferenciarlos bien para poder pasar con soltura de uno a otro.

De manera paralela a estos aprendizajes derivados directamente del desarrollo de los objetivos del TFM, también se han obtenido nuevos conocimientos en el área del trabajo colaborativo y de control de versiones con la plataforma github o la redacción de un texto técnico como es una memoria de final de máster en formato LaTeX. La suma de todos estos nuevos conocimientos adquiridos, serán de mucha utilidad en otros ámbitos.

Los objetivos incialmente planteados incluían la interpretación de los resultados obtenidos en las cepas seleccionadas, pero hubo que prescindir de esa fase y alterar la planificación del trabajo a desarrollar. Esto ha sido debido en gran medida a fallos en la estimación del tiempo requerido para el desarrollo de los primeros módulos del proceso. No se tuvo en cuenta el tiempo requerido para la familiarización con la metodología, el aprendizaje profundo de un lenguaje de programación y la búsqueda de información para resolver los errores que se generaban debido a un código poco estable. Tampoco se consideró inicialmente la posibilidad de que los datos con los que se iban probando los paquetes no presentaran siempre la misma estructura o tipo de información que los que se encuentran en las bases de datos o generen *de novo* el usuario. Esto produjo una continua necesidad de depurar y adaptar el código a las nuevas variables que iban apareciendo.

Aún así, se ha obtenido una herramienta funcional capaz de tomar cualquier archivo en los formatos compatibles y extraer la información necesaria para llevar a cabo exitosamente la búsqueda de genes duplicados, relacionarlos entre sí y representar de manera atractiva toda esta información. Esta herramienta podrá ser útil para la comunidad científica y avanzar en la comprensión de los mecanismos bacterianos. Especialmente cuando se da el caso de que son las bacterias más interesantes a nivel clínico por su virulencia y resistencia a antibióticos las que presentan mayores tasas de duplicación genética. Hasta ahora, el estudio de duplicidades génicas requerían de la continua intervención del investigador para pasar de una fase a la siguiente, con esta herramienta se permitirá realizar todo el proceso de manera automática y generar resultados con los que trabajar.

Como principal línea de trabajo futuro, claramente habría que conectar el módulo de R con el programa en python y permitir que se pueda descargar directamente los archivos de anotación de las bases de datos para terminar de automatizar todo el trabajo. También habría que incluir el código necesario para poder trabajar a gran escala con grandes cantidades de anotaciones diferentes y depurar en profundidad todo el proceso para mejorar la herramienta desarrollada.

Resultaría interesante complementar los trabajos previos en duplicidades génicas en bacterias del grupo ESKAPE y localizar e identificar genes duplicados con particularidades relevantes. La inclusión de un método de análisis de elementos genéticos móviles y de genes asociados a virulencia y resistencia antimicrobiana junto con la caracterización de posibles correlaciones y patrones de duplicación terminaría de completar los temas que se plantearon inicialmente a la hora de establecer los objetivos del TFM.

Finalmente, sería de gran utilidad divulgativa realizar una guía completa de uso del programa para favorecer la comprensión de su manejo y los resultados obtenidos.

# Glosario

***Acinetobacter baumannii*** Bacteria gram-negativa aeróbica. Puede formar parte de la flora epitelial y colonizar las vías respiratorias y digestivas y causar neumonía severa e infecciones del tracto urinario.

**Anotación del genoma** Identificación y asignación de funciones a los distintos elementos presentes en la secuencia genética de un organismo.

**Biocircos** Libreria gráfica de R para visualizar datos genómicos.

**BLAST** Basic Local Alignment Search Tool. Es una herramienta informática de alineamiento de secuencias. Es capaz de comparar una secuencia problema (*query* ) con las secuencias que se encuentren en una base de datos.

**Cadena de procesos** Serie de procesos, generalmente lineales y unidireccionales, que toman unos datos de entrada y los transforma en datos de salida. El primer proceso toma los datos sin procesar como entrada, realiza una serie de acciones sobre ellos y envía los resultados al siguiente proceso. Termina con el resultado final producido por el último proceso de la línea. Por definición, cada uno de los procesos es autónomo, se pueden ejecutar fuera de la cadena.

**CDS** Coding Sequence o región de codificación de un gen.

**Duplicación genética** Duplicación de una región del genoma que engloba al menos un gen.

**Ensamblaje del genoma** Proceso mediante el cual se representan los cromosomas originales de la secuencia genómica de un organismo a partir de los múltiples fragmentos generados durante su secuenciación.

***Enterobacter* spp.** Género de bacterias gram-negativas anaerobias facultativas. Muchas de las especies del género son patógenas y causa de infecciones oportunistas como el resto de integrantes del grupo ESKAPE. Pueden provocar infecciones en el tracto urinario y respiratorio.

**ESKAPE** Acrónimo que engloba a las especies bacterianas patógenas *Enterococcus spp, Staphylococcus aureus, Klebsiella pneumoniae, Acinetobacter baumanni, Pseudomonas aeruginosa* y *Enterobacter* spp. , que se caracterizan por ser especialmente virulentas al presentar diversos genes de resistencia a antimicrobianos.

**e-valor** Valor de significancia de los alineamientos obtenidos por BLAST teniendo en cuenta la probabilidad de que obtengan la misma puntuación por azar

**FASTA** formato de fichero basado en texto, utilizado para representar secuencias. Comienza con una descripción en una única línea comenzada por el símbolo >, seguida por líneas de datos de secuencia. La simplicidad del formato lo hace muy sencillo de manipular y analizar.

**Flujo de trabajo** Conjunto de procesos que filtran o transforman datos. Puede ramificarse o ser lineal. Generalmente no hay un "primer" proceso claramente definido: los datos pueden ingresar al flujo de trabajo en múltiples puntos. Cualquier proceso podría tomar datos sin procesar como entrada y enviar sus resultados a otro proceso o generar un "resultado final".

**GenBank** Base de datos de secuencias genéticas del NIH (National Institutes of Health de Estados Unidos) de disponibilidad pública.

**Genes de resistencia** Expresión genética que permite a una bacteria sobrevivir en un ambiente con presencia de un antibiótico específico.

**GFF** General Feature Format. Formato para la descripción de los componentes de secuencias genómicas. Delimitados por tabulaciones con nueve campos por línea.

**Hit** Cada una de las secuencias similares obtenidas al realizar el alineamiento de secuencias a estudio con secuencias de referencia. Homología Situación en la que dos o más secuencias presentan un alto grado de similitud por lo que se deduce una relación ancestral común.

***Klebsiella pneumoniae*** Bacteria gram-negativa anaerobia facultativa ampliamente distribuida en el ambiente. Puede colonizar las vías nasofaríngeas y el tracto gastrointestinal provocando neumonías e infecciones urinarias. Es una especie muy frecuente en entornos hospitalarios que puede provocar infecciones graves en neonatos o pacientes de postoperatorio. mutación adaptativa mutaciones que aumentan el éxito evolutivo y se transmiten de un organismo a otro perdurando en el tiempo.

**NCBI** National Center of Biotechnology Information. Es es parte de la Biblioteca Nacional de Medicina de Estados Unidos y almacena y actualiza constantemente la información referente a secuencias genómicas. También ofrece herramientas bioinformáticas para el análisis genómico a diferentes niveles y un índice de los artículos biomédicos en investigación. Estas bases de datos están disponibles en linea de forma gratuita. patrones de expresión forma de expresarse característica de un conjunto se secuencias, indicando qué posiciones son más importantes y cuales pueden modificarse y cómo. Determinar patrones de expresión de proteínas es clave para determinar su función o estructura.

**Pipeline** ver cadena de procesos

**Plásmidos** Moléculas circulares de material genético extracromosómico, en bacterias y levaduras, capaces de replicarse de manera autónoma y transmitirse entre organismos.

**Pseudogen** Copia de un gen ya conocido y de función distinta que pueden haber modificado su funcionalidad o haberla cambiado completamente debido a la falta de intrones y otras secuencias de ADN esenciales para su función. Aunque son genéticamente similares al gen funcional original, no se expresan y suelen presentar numerosas mutaciones

***Pseudomonas aeruginosa*** Patógeno oportunista perteneciente al grupo de las bacterias gram-negativas aeróbicas. Infecta los pulmones y vías respiratorias provocando neumonías de caracter grave. También pueden infectar las vías urinarias o tejidos.

**Python** Lenguaje de programación desarrollado como proyecto de código abierto. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo.

**R** Lenguaje de programación para computación estadística que permite desarrollar gráficos de alto nivel.

**Script** Término coloquial a la secuencia de comandos de un programa sencillo

**Transposón** Secuencia de material genético capaz de desplazarse a diferentes partes del genoma de una célula, pudiendo causar mutaciones en el proceso.

**Workflow** ver flujo de trabajo

# Bibliografía


[1] Altschul, S. F. / Gish, W. / Miller, W. / Myers, E. W. / Lipman, D. J.(1990): *Basic local alignment search tool*, 3: 403–410.

[2] Romero, David / Palacios, Rafael(1997): *Gene amplification and genomic plasticity in prokaryotes*, 1: 91–111.

[3] Murray, B. E.(2000): *Vancomycin-resistant enterococcal infections*, 10: 710–721.

[4] Zhang, Jianzhi(2003): *Evolution by gene duplication: an update*, 6: 292–298.

[5] Lynch, Michael u.a.(2008): *A genome-wide view of the spectrum of spontaneous mutations in yeast*, 27: 9272–9277.

[6] Serres, Margrethe H. / Kerr, Alastair RW / McCormack, Thomas J. / Riley, Monica(2009): *Evolution by leaps: gene duplication in bacteria*, 1: 46.

[7] Toro, Lina María Echeverri / Correa, Juan Carlos Cataño(2010): *Klebsiella pneumoniae como patógeno intrahospitalario: epidemiología y resistencia*, 3: 11.

[8] Lipinski, Kendra J. / Farslow, James C. / Fitzpatrick, Kelly A. / Lynch, Michael / Katju, Vaishali / Bergthorsson, Ulfar(2011): *High Spontaneous Rate of Gene Duplication in Caenorhabditis elegans*, 4: 306–310.

[9] Pendleton, Jack N. / Gorman, Sean P. / Gilmore, Brendan F.(2013): *Clinical relevance of the ESKAPE pathogens*, 3: 297–308.

[10] Fariñas, María Carmen / Martínez Martínez, Luis(2013): *Infecciones causadas por bacterias gramnegativas multirresistentes: enterobacterias, Pseudomonas aeruginosa, Acinetobacter baumannii y otros bacilos gramnegativos no fermentadores*, 6: 402–409.

[11] Reams, Andrew B. / Roth, John R.(2015): *Mechanisms of gene duplication and amplification*, 2: a016592.

[12] Cui, Ya u.a.(2016): *BioCircos.js: an interactive Circos JavaScript library for biological data visualization on web applications*, 11: 1740–1742.

[13] Santajit, Sirijan / Indrawattana, Nitaya(2016): *Mechanisms of Antimicrobial Resistance in ESKAPE Pathogens.*

[14] Bernabeu, Manuel / Sánchez Herrero, José Francisco / Huedo, Pol / Prieto, Alejandro / Hüttener, Mário / Rozas, Julio / Juárez, Antonio(2019): *Gene duplications in the E. coli genome: common themes among pathotypes*, 1: 313.

[15] Sanchez Herrero, José Francisco / Bernabeu, Manuel / Prieto, Alejandro / Hüttener, Mário / Juárez, Antonio(2020): *Gene Duplications in the Genomes of Staphylococci and Enterococci.*

[16] Chávez Jacobo, Víctor M.(2020): *La batalla contra las superbacterias: No más antimicrobianos, no hay ESKAPE*, 0: .

[17] Chen, Xiaowei / Cui, Ya (2019): *BioCircos: Interactive Circular Visualization of Genomic Data using 'htmlwidgets' and 'BioCircos.js'*
, R package version 0.3.4 https://CRAN.R-project.org/package=BioCircos.

[18] Biotechnology Information, National Center for / Medicine, Bethseda (MD): U. S. National Library of (1988): *National Center for Biotechnology Information*
https://www.ncbi.nlm.nih.gov/ (Accedido 02/01/2021), USA.

[19] Biotechnology Information (US), Bethesda (MD): National Center for (2008): *BLAST® Command Line Applications User Manual.* , National Center for Biotechnology Information (US).

[20] Grimont, Francine / Grimont, Patrick A. D. (2006): *The Genus Enterobacter.* In: Dworkin, Martin / Falkow, Stanley / Rosenberg, Eugene / Schleifer, Karl Heinz / Stackebrandt, Erko (Hg.), The Prokaryotes.Springer New York: 197–214.

[21] Madden, Tom (2003): *The BLAST Sequence Analysis Tool.* , National Center for Biotechnology Information (US).

[22] *Home - Genome - NCBI*
https://www.ncbi.nlm.nih.gov/genome/ (Accedido 05/01/2021).

[23] Medicine (US), Bethesda (MD): National Library of / Biotechnology Information, National Center for *BLAST: Basic Local Alignment Search Tool*
https://blast.ncbi.nlm.nih.gov/Blast.cgi (Accedido 03/01/2021).

[24] Scholz, Matthias (2020): *BLASTn output format 6 - Metagenomics*
http://www.metagenomics.wiki/tools/blast/blastn-output-format-6 (Accedido 04/01/2021).

[25] Scholz, Matthias (2020): *E-value & Bit-score - Metagenomics*
http://www.metagenomics.wiki/tools/blast/evalue (Accedido 04/01/2021).

[26] Team, R Core (2020): *R: A Language and Environment for Statistical Computing*
.

[27] Todar, Kenneth (2020): *Pseudomonas*
http://textbookofbacteriology.net/pseudomonas.html (Accedido 05/01/2021).

[28] Van Rossum, Guido / Drake, Fred L. (2009): *Python 3 Reference Manual*. Scotts Valley, CA, CreateSpace.

[29] Vulliard, Loan (2019): *BioCircos: Generating circular multi-track plots*
https://cran.r-project.org/web/packages/BioCircos/vignettes/BioCircos.html#session-info (Accedido 05/01/2021).

# Apéndice A

# Requerimientos y código

Para la realización de este TFM se ha trabajado íntegramente en un entorno Linux por lo que los comandos de instalación utilizados han sido en este lenguaje desde la consola.

## A.1. *python*

Los módulos instalados en el ambiente *python* generado específicamente para este TFM se pueden encontrar en la siguiente url al repositorio de github https://github.com/albamgarces/TFM_UOC_AMoya/tree/main/memoria/requeriments

**Paquetes** *pip*: Los siguientes módulos y respectivas versiones fueron instalados para el correcto funcionamiento del programa desarrollado con el siguiente comando:

```
$ pip install nombre_programa
```

- bcbio-gff 0.6.6: Librería que permite leer archivos GFF
- biopython 1.78: Conjunto de aplicaciones y programas con aplicaciones bioinformáticas.
- ftputil 4.0.0: Librería que permite acceder a servidores FTP.
- numpy 1.19.2: Paquete que permite crear vectores y matrices multidimensionales grandes y provee de funciones matemáticas de alto nivel para trabajar con ellas.
- pandas 1.1.2: Biblioteca que en combinación con NumPy permite manipular los datos y crear tablas o series temporales.
- wget 3.2: herramienta que permite la descarga de contenidos desde servidores web.

**BLAST**: `$ sudo apt-get install ncbi-blast+` Con este comando se descarga la última versión de BLAST, si se prefiere descargar otra versión diferente, NCBI dispone de una carpeta ftp desde la que se puede descargar una versión específica: ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/

Una vez instalado, se puede utilizar directamente desde la terminal. Para generar una base de datos, utilizaremos los siguientes comandos:

```
$ makeblastdb -in fasta_file -input_type fasta -dbtype prot -out name
```

Donde *fasta_file* es el fichero de proteínas; *name* es el nombre que le daremos a la nueva base de datos; *input_type* es el formato del fichero proporcionado; y *dbtype* es el tipo de secuencias proporcionadas (en nuestro caso, proteínas).

Una vez generada la base de datos, se podrá hacer la búsqueda de alineamientos con blastp con el siguiente comando:

```
$ blastp -query fasta_file -db name -outfmt '6 std qlen slen' -num_threads X -out name_out
```

Donde *query* son las secuencias problema; *num_threads* es el número de CPUs que se utilizarán (una, en nuestro caso); y *outfmt* es el tipo de formato que queremos darle a nuestra tabla de salida. En este caso, se elige el formato 6 con las columnas estándar y además se añaden las columnas de longitud de las secuencias problema y de referencia. Se puede encontrar más información sobre el formato 6 en [24]

Todos los scripts de python para este TFM se pueden encontrar en la siguiente url de github: https://github.com/albamgarces/TFM_UOC_AMoya/tree/main/memoria/scripts

## A.2. $\mathbb{R}$

A continuación se muestra la información de la sesión de R utilizada para la generación de gráficos circulares con los resultados de python:

```
> sessionInfo()
R version 4.0.3 (2020-10-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.1 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0

locale:
 [1] LC_CTYPE=es_ES.UTF-8       LC_NUMERIC=C               LC_TIME=es_ES.UTF-8
 [4] LC_COLLATE=es_ES.UTF-8     LC_MONETARY=es_ES.UTF-8    LC_MESSAGES=es_ES.UTF-8
 [7] LC_PAPER=es_ES.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=es_ES.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] dplyr_1.0.2      colorspace_2.0-0 BioCircos_0.3.4

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.5         magrittr_2.0.1     tidyselect_1.1.0   R6_2.5.0
 [5] rlang_0.4.9        fansi_0.4.1        plyr_1.8.6         tools_4.0.3
 [9] cli_2.2.0          htmltools_0.5.0    ellipsis_0.3.1     yaml_2.2.1
[13] digest_0.6.27      assertthat_0.2.1   tibble_3.0.4       lifecycle_0.2.0
[17] crayon_1.3.4       purrr_0.3.4        RColorBrewer_1.1-2 htmlwidgets_1.5.3
[21] vctrs_0.3.6        glue_1.4.2         compiler_4.0.3     pillar_1.4.7
[25] generics_0.1.0     jsonlite_1.7.2     pkgconfig_2.0.3
```

El script completo para la creación de BioCircos con los datos generados en dup_annot.py se puede encontrar en el siguiente enlace: https://github.com/albamgarces/TFM_UOC_AMoya/blob/main/memoria/scripts/dup_biocircos.R

# Apéndice B

# Análisis de duplicaciones génicas en cepas seleccionadas

Se puede encontrar el análisis completo para todas las cepas así como una tabla resumen englobando la información contenida en NCBI y los resultados obtenidos en la búsqueda de duplicados en el siguiente enlace:

https://github.com/albamgarces/TFM_UOC_AMoya/tree/main/testing

Cuadro B.1: Tabla resumen de los resultados obtenidos al realizar el proceso de análisis de duplicados con dup_annot.py en slas cepas seleccionadas.

| Especie | Cepa | Total proteinas | Grupos | Duplicados |
|---|---|---|---|---|
| *Klebsiella pneumoniae* | KpC11 | 5472 | 152 | 270 |
| | ATCC 43816 | 5485 | 124 | 178 |
| | Beach Ranger | 5040 | 109 | 129 |
| | KpPF25 | 5222 | 102 | 137 |
| *Acinetobacter baumannii* | ATCC 17961 | 3783 | 122 | 219 |
| | TP3 | 3571 | 71 | 129 |
| | TP2 | 3572 | 69 | 124 |
| | FDAARGOS_1036 | 3808 | 116 | 205 |
| *Pseudomonas aeruginosa* | DL201330 | 6046 | 116 | 161 |
| | TJ2014-049 | 6089 | 102 | 158 |
| | TJ2019-017 | 5904 | 63 | 93 |
| | TJ2019-022 | 6686 | 191 | 333 |
| *Enterobacter cloacae* | STN0717-73 | 5022 | 92 | 240 |
| | STN0717-60 | 4522 | 31 | 242 |
| | RHBSTW-00399 | 5044 | 156 | 242 |
| | RHBSTW-00490 | 5086 | 157 | 261 |

A continuación se incluyen los gráficos generados con BioCircos para otras cepas de *K. pneumoniae* seleccionadas. El cromosoma de la bacteria (rojo) y sus plásmidos (azul) se representan en un mapa circular que muestra la localización de cada gen duplicado. Las siguientes dos capas muestran los genes en la hebra positiva (rosa) y negativa (turquesa). Las líneas naranja interiores señalan las conexiones entre proteínas pertenecientes al mismo grupo de duplicados. Escala de tamaño en Mb

Figura B.1: Representación gráfica con BioCircos de los genes duplicados en el genoma de la cepa ATCC43816 de *K. pneumoniae*.

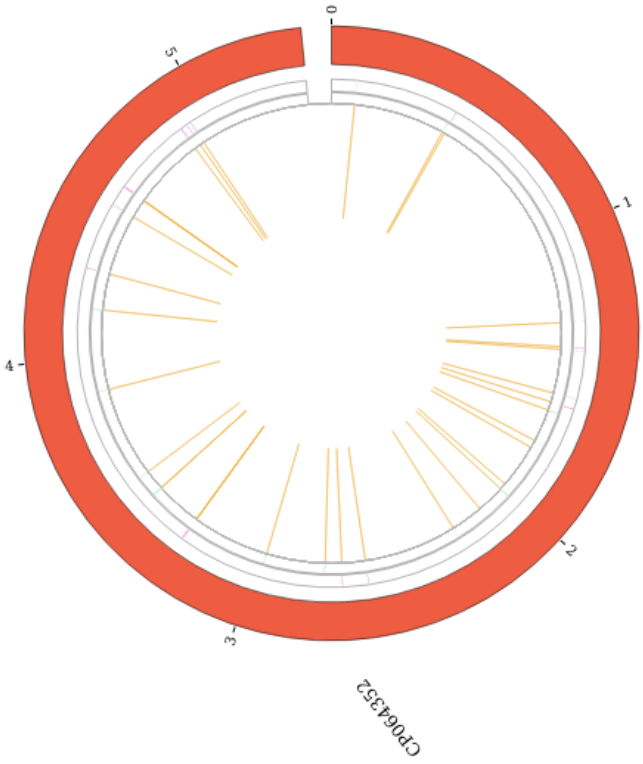

Figura B.2: Representación gráfica con BioCircos de los genes duplicados en el genoma de la cepa KpC11 de *K. pneumoniae*.

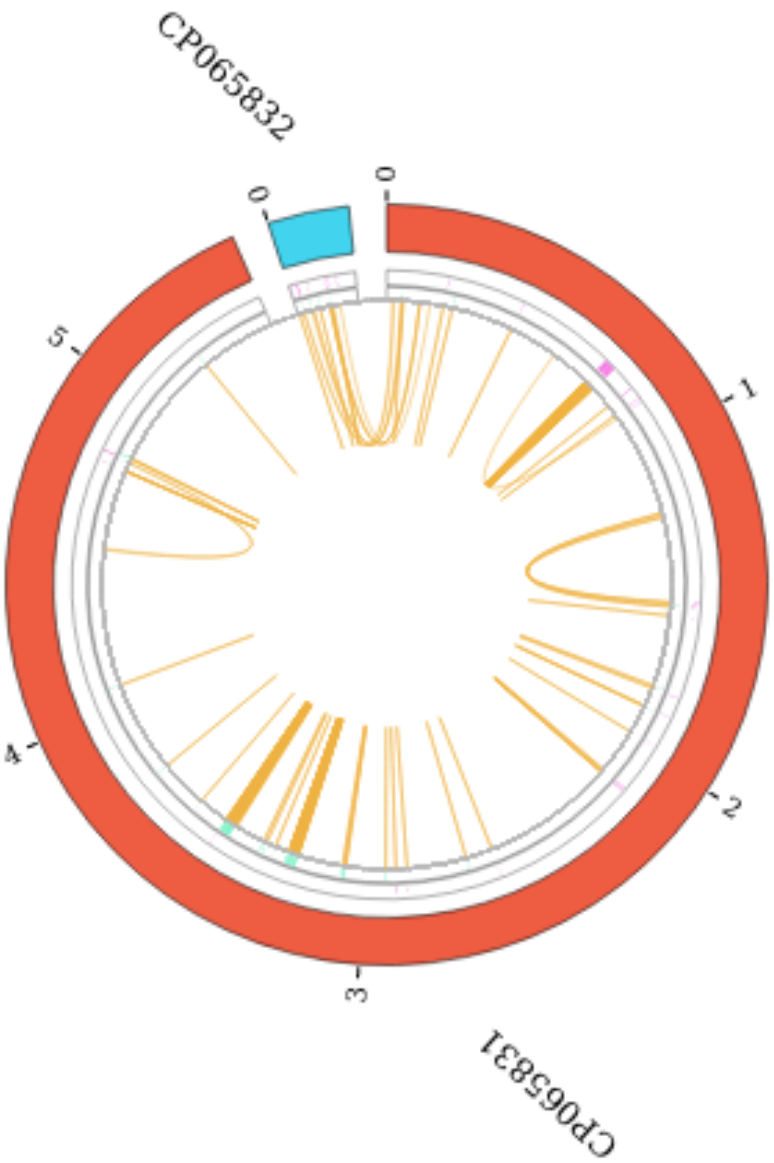

Figura B.3: Representación gráfica con BioCircos de los genes duplicados en el genoma de la cepa KpPF25 de *K. pneumoniae*.